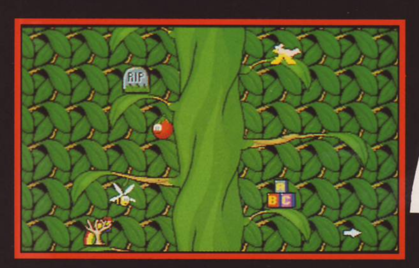


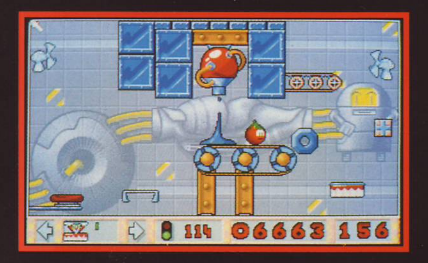
SONDERHEFT 87
Markt&Technik
öS 120,-/sfr 16,-/Lit 18000
hfl 21,-/dkr 75,-/fmk 72,-
DM 16,-
DISKETTE IM HEFT
64'er
36 Programme auf Diskette
GRAFIK
Grafik-Komfort
85 neue, blitzschnelle Grafikbefehle
Intros & Levels
2 ultimative Tools für Vorspänne und Spiele-Screens
Tips, Tricks & Utilities
Raffinierte Routinen für Hires und Multicolor
Grafik-Klau
Picture-Tool V1.0 spürt Bilder und Zeichensätze auf

Bill's TOMATO GAME™

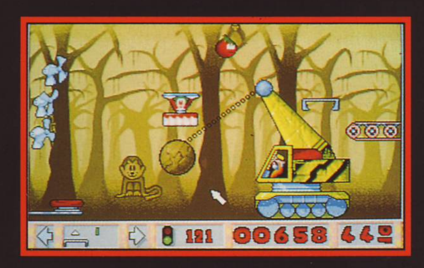
Bunter als ein Picasso!
Süßer als Schokoladeneis!!
Frecher als alle anderen Früchtchen!!!
Machen Sie die Bekanntschaft von T'n'T, einem ganz gewöhnlichen Tomatenpärchen, und erleben Sie eine explosive Mischung von wahrer Liebe und (ent) fesselnden Bildschirm-Orgien mit mehreren Ventilatoren.
Ketch-up!





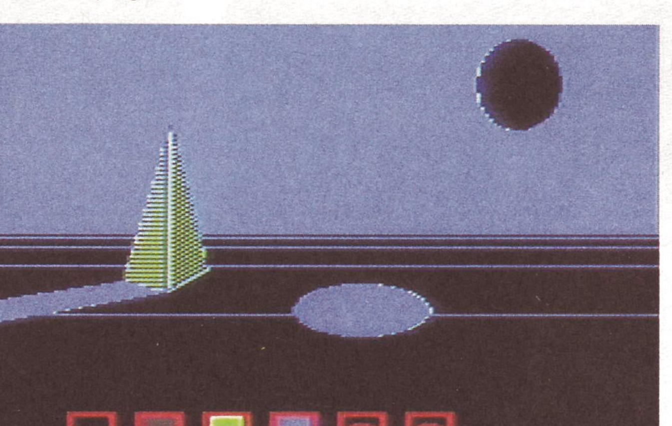




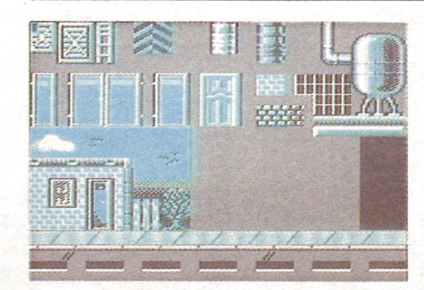


Deutsche Vertretung: Alliance Division, Verlerstr.1, 4830 Gütersloh





Picture Tool 1.0: Der Grafikspürhund stöbert jede Grafik auf: ein Demo-Bild im Speicher ab $6000 Seite 44



Game-Level-Designer V2.0: Überdimensionale Spielelandschaften – 256 Zeichen breit und 25 Zeilen hoch Seite 45

Alle Programme aus Artikeln mit einem Disketten-Symbol finden Sie auf der beiliegenden Diskette (Seite 19).

Grundlagen – Demos selbst programmiert

Programmieren wie ein Profi

Demo-Entwickler sind keine Übermenschen und haben auch keinerlei überirdische Begabung. Warum diese Programmierer auch nur mit Wasser kochen, erfahren Sie in unseren Grundlagen zur Demoprogrammierung.

Bevor Sie weiterlesen, sollten Sie allerdings drei Fragen ohne Zögern ehrlich mit »JA« beantworten können:
1. Kennen Sie sich in Ihrem Assembler aus?
2. Können Sie mit Ihrem Maschinenmonitor umgehen?
3. Beherrschen Sie die Maschinensprache?

Ohne diese Grundvoraussetzungen werden Sie mit den folgenden drei Seiten nicht allzuviel anfangen können. Aber keine Panik: selbst wenn Sie Ihren Monitor oder Assembler nicht ganz genau kennen, Ausprobieren lohnt sich in jedem Fall.

Ideenfindung

Am Anfang jeder Demo steht eine Idee. Ob diese die Demo-Szene revolutionieren wird oder nur mit dem Strom mitschwimmt ist im Grunde egal. Es steht Ihnen ohnehin viel Programmierarbeit ins Haus. Wir wollen die Programmierung einer einfachen Demo anhand eines Beispiels (Abb. 1) verdeutlichen: Raster-Bar-Scrolling, Color-Cycling, Flashing Chars und Smoothscrolling gibt's nach Laden und Starten per RUN zu bestaunen.

Grundlage jeder Demo sollte folgendes Programmgerüst sein:
1) Variablen-Deklaration
2) IRQ-Vorbereitung
3) IRQ-Hauptschleife
4) Effekt(Sub)-Routinen
5) Initialisierungs-Routinen

Die Variablendeklaration (Symbol-Tabelle) ist zwar nicht unbedingt nötig, hilft Ihnen aber, wenn es z.B. um das Ändern einer Adresse geht. Statt den ganzen Source-Code nach der Adresse zu durchforsten, genügt es, das Byte im Deklarationsteil zu ändern.



[1] Smoothscrolling, Raster-Bars und Charflasher selbst programmiert

```
;----------------------------------------
;--------VARIABLEN SETZEN----------------
;----------------------------------------

IRQLOW     = $0314  ;IRQ-VEKTOR LOWBYTE
IRQHIG     = $0315  ;           HIGHBYTE
OLDIRQ     = $EA31  ;ALTE IRQ-ROUTINE

INITMUSIC  = $1000  ;MUSIK INITIALISIEREN
PLAYMUSIC  = $1003  ;ABSPIELEN

FLASHCOUNT  = $FB   ;ZAEHLER 1 UND 2
FLASHCOUNT2 = $FC   ;FUER FLASH-EFFEKTE

SCRHELP    = $03FB  ;HILFSREGISTER F.
SCROLLREG  = $D016  ;SCROLLREGISTER

CHARSET    = $D018  ;ZEICHENS. DEFINIEREN
RASTER     = $D012  ;RASTERSTRAHL-POS
YSCROLL    = $D011  ;Y-SCROLL-REGISTER
IMR        = $D01A  ;IRQ MASK REGISTER
```

So oder so ähnlich könnte auch Ihre Symbol-Tabelle aussehen. Sinnvolll sind in jedem Fall klare und präzise Label-Namen, die die Funktion der Speicherstelle erklären, nicht verschleiern. So weist beispielsweise der Label (oder das









[2] FLD, Splitted Screen, Rastersprites und ein Wassereffekt sind schwierig zu programmieren

Symbol) »SCROLLREG« eindeutig auf das im VIC vorhandene Hardscroll-Register $D016 hin, das Symbol »PLAYMUSIC« auf die Player-Routine der Musik usw. Bei knappem Speicher – und das ist eigentlich immer gegeben – sind kurze Namen aber unerläßlich, denn je kürzer der Name, um so mehr Labels passen in den Speicher.

Nach unserer Symboltabelle geht's ans Eingemachte:

```
;----------------------------------------
;--------IRQ VORBEREITEN-----------------
;----------------------------------------

        SEI           ;IRQ SPERREN

        JSR INIT      ;INITIALISIEREN

        LDA #<START   ;LOWBYTE LADEN
        STA IRQLOW    ;UND SPEICHERN
        LDA #>START   ;HIGHBYTE LADEN
        STA IRQHIG    ;UND SPEICHERN

        LDA YSCROLL   ;Y-SCROLL-REGISTER
```

```
AND #$7F     ;RICHTIG
STA YSCROLL  ;SETZEN

LDA #$7F     ;TIMER
STA $DC0D    ;SETZEN

LDA #$01     ;RASTER-IRQ
STA IMR      ;FESTLEGEN

LDA #$00     ;MUSIK
JSR INITMUSIC;INITIALISIEREN
CLI          ;IRQ FREIGEBEN
JMP *        ;ENDLOS-SCHLEIFE
```

Zunächst müssen wir die IRQ-Vektoren »verbiegen«, das Y-Scroll-Register bzw. die Musik-Routine initialisieren und alle möglichen Bildschirm-Masken setzen (JSR INIT). In dieser Routine löschen wir den Bildschirm, setzen diverse Variablen und schreiben die Farben (für's spätere Farbscrolling) und Texte in den Farb- bzw. Bildschirmspeicher:

```
;---------------------------------------
INIT    LDA #$C7         ;HILFSREGISTER
        STA SCRHELP      ;SETZEN

        LDA #$00         ;COUNTER
        STA FLASHCOUNT   ;INITIALISIEREN
        STA FLASHCOUNT2;
        STA $D020        ;SCREEN UND
        STA $D021        ;FRAME SCHWARZ
        LDA #$00         ;SCHWARZES
        STA $0286        ;FARBRAM
        JSR $E544        ;SCREEN LOESCHEN

        LDX #$00         ;"DEMO TEST
INIT1   LDA SCREEN1,X    ; GROSSER
        STA $0454,X      ; ZEICHENSATZ"
        INX              ;AUF DEN
        CPX #$EC         ;BILDSCHIRM
        BNE INIT1        ;SCHREIBEN

        LDX #$77         ;"KLEINER
INIT3   LDA SCREEN2,X    ; ZEICHENSATZ"
        STA $0608,X      ;AUF DEN
        DEX              ;BILDSCHIRM
        BPL INIT3        ;SCHREIBEN

        LDX #$27         ;FARBTABELLE
INIT4   LDA FLASHTAB2,X;IN COLOR-RAM
        STA $DB20,X      ;SCHREIBEN
        DEX              ;(FUER'S
        BPL INIT4        ;FARBSCROLLING)

        LDA #$C7         ;SCROLLREGISTER
        STA SCROLLREG    ;SETZEN
        LDA #$1C         ;ZEICHENSATZ
        STA CHARSET      ;ANSCHALTEN
        RTS              ;ENDE
```

Das Herz jeder Demo ist die IRQ-Hauptschleife. Hier passiert alles, was Sie später auf dem Bildschirm bewundern können. Die einzelnen Effekt-Routinen werden innerhalb dieser Schleife per JSR aufgerufen und abgearbeitet. Da wir den Anschein erwecken wolle, Grafik-Effekte und Musik liefen gleichzeitig ab, ist es dringend notwendig, daß es sich bei unseren JSR-Routinen um sog. Single-Step-Procedures handelt. Rufen wir beispielsweise mit JSR PLAYMUSIC die Player-Routine des Musikstücks auf, darf diese unter keinen Umständen erst nach komplettem Abspielen sämtlicher Noten zurückkehren, sondern muß bei Aufruf die jeweils aktuelle Note spielen und danach sofort per RTS unterbrechen. Jetzt kommt sofort die Raster-Routine an die Reihe usw. Durch dieses Timesharing – was z.B. auch sog. Multi-Tasking-Computer nutzen – und der Schnelligkeit unseres MOS6510-Prozessors entsteht der Eindruck eines gleichzeiten Ablaufs mehrerer Programme. Apropos Musik: keine Bange vor Musikuntermalung, das gehört mit Sicherheit zu den einfachsten Dingen in einer Demo: Da die modernen Composer in jedem Stück über eine eigene Player- und Init-Routine verfügen, müssen wir später diese beiden Routinen nur per JSR $XXXX anspringen. Die beiden Adressen lassen sich mit einem Monitor schnell herausfinden. Charakteristisch sind z.B.

Future-Composer $1000 (INIT) $1006 (PLAY) oder
Future-Composer $1800 (INIT) $1806 (PLAY)
JCH-Player $1000 (INIT) $1003 (PLAY)
MAC-Player $1048 (INIT) $1021 (PLAY)
RoMuzak $8000 (INIT) $8003 (PLAY)

Zurück zur Hauptschleife:

```
;---------------------------------------
;--------IRQ-HAUPTSCHLEIFE--------------
;---------------------------------------
START   LDA YSCROLL     ;WARTEN BIS
        BPL START       ;UNTERER RAND
                        ;ERREICHT
        LDA #$1A        ;ZEICHENSATZ
        STA CHARSET     ;UMSCHALTEN
        LDA #$C8        ;SCROLLREG
        STA SCROLLREG   ;SETZEN

        JSR PLAYMUSIC   ;MUSIK SPIELEN

        LDA #$38        ;AUF RASTERLINE
ST1     CMP RASTER      ;$38
        BNE ST1         ;WARTEN

        JSR RASTERSHOW;RASTER-FARBEN
        JSR SCROLL      ;FARBEN SCROLLEN

        LDA #$8C        ;AUF RASTERLINE
ST2     CMP RASTER      ;$8C
        BNE ST2         ;WARTEN

        LDA #$1C        ;KLEINEN ZEICHEN
        STA CHARSET     ;SATZ EIN

        JSR FLASH       ;3 LINIEN FLASHEN

        LDA #$B8        ;AUF RASTERLINE
ST3     CMP RASTER      ;$B8
        BNE ST3         ;WARTEN

        JSR CHARSCROLL;SCROLLROUTINE
        JSR CHARFLASH   ;FARBSCROLLING

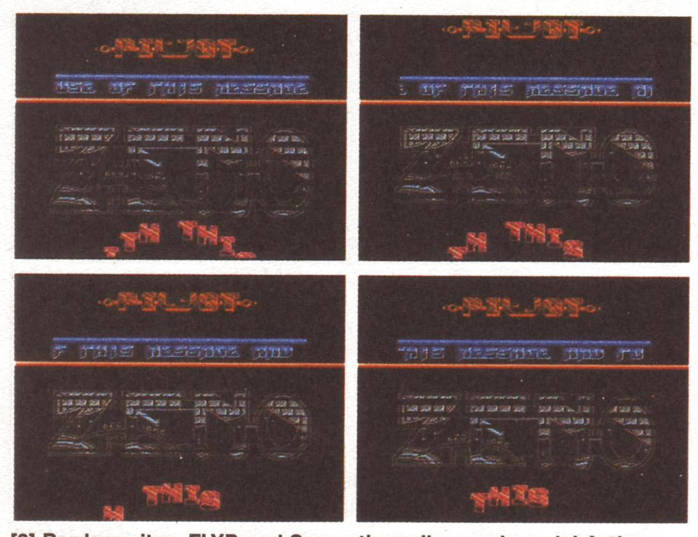
        JMP OLDIRQ      ;ALTER IRQ
```

Auffallend sind zunächst die »STx«-Schleifen. Diese warten auf eine bestimmte Rasterzeile und starten danach eine bestimmte Aktion. Ab Rasterzeile $38 (oberer sichtbarer Bildschirmrand) starten wir unsere Rasterbalken-Anzeige (auch Copper-Bars, Coppers oder Raster-Bars genannt), danach, ein paar Rasterzeilen später unsere Color-Cycling-Routine und schließlich unsere Laufschrift. Wir teilen den Bildschirm also in kleine oder große Bereiche ein, in denen verschiedene Aktivitäten ablaufen. Wie bereits erwähnt kommt am Anfang die Raster-Subroutine an die Reihe:

```
RASTERSHOW
        LDX #$00        ;ZAEHLER AUF $00
COLOR1  LDA COLTAB,X;FARBWERT HOLEN
        LDY WAITAB,X;WAIT-WERT HOLEN
WAIT1   DEY             ;HERUNTERZAEHLEN
        BNE WAIT1       ;UM DIE ZYKLEN
                        ;AUSZUGLEICHEN
```

```
        STA $D021     ;FARBE
        STA $D021     ;SCHREIBEN
        INX           ;ZAEHLER+1
        CPX #$40      ;SCHON 40 FARBEN?
        BNE COLOR1    ;NEIN DANN COLOR1
        LDA #$00      ;JA DANN SCREEN
        STA $D020     ;AUF SCHWARZ
        STA $D021     ;SETZEN
        RTS           ;ENDE

COLTAB
        .BYTE $06,$04,$0E,$03,$07,$0F
        .BYTE $01,$01,$0F,$07,$03,$0E
        .BYTE $06,$04,$00,$00,$09,$02
        .BYTE $0A,$07,$0F,$01,$01,$0F
        .BYTE $07,$0A,$02,$09,$00,$00
        .BYTE $09,$0B,$08,$0C,$0F,$07
        .BYTE $01,$01,$0F,$0C,$08,$0B
        .BYTE $09,$00,$00,$02,$0A,$07
        .BYTE $0F,$01,$0B,$0C,$0F,$01
        .BYTE $01,$0F,$0C,$0B,$01,$0F
        .BYTE $07,$0A,$02,$00,$00
COLEND
WAITAB
        .BYTE $09,$08,$08,$01,$08,$08
        .BYTE $08,$08,$08,$08,$08,$01
        .BYTE $08,$08,$08,$08,$08,$08
        .BYTE $08,$01,$08,$08,$08,$08
        .BYTE $08,$08,$08,$01,$08,$08
        .BYTE $08,$08,$08,$08,$08,$01
        .BYTE $08,$08,$08,$08,$08,$08
        .BYTE $08,$01,$08,$08,$08,$08
        .BYTE $08,$08,$08,$01,$08,$08
        .BYTE $08,$08,$08,$08,$08,$01
        .BYTE $08,$08,$08,$08,$08
```



[3] Bordersprites, FLYP und Scmoothscroller machen viel Action

Störungsfreie eingefärbte Bildschirmzeilen sind damit kein Problem. In der ersten Tabelle (»COLTAB«) stehen die Farbwerte, in der zweiten (»WAITAB«) die Verzögerungs-Bytes, die das genaue Timing ermöglichen. Das Prinzip ist einfach: wir schreiben eine Bildschirmfarbe ($D021) und warten so lange, bis der Rasterstrahl die gesamte Bildschirmbreite abgetastet hat. Danach kommt sofort die nächste Farbe und das nächste Wait-Byte an die Reihe usw. Sie werden sich vielleicht wundern, warum sich jedes achte Byte in der Wait-Tabelle von den anderen unterscheidet (durch die kleinere Wertigkeit wird natürlich auch die Verzögerungsschleife wesentlich schneller abgearbeitet). Ganz einfach: der VIC braucht jede achte Rasterzeile, um die nächste Charakter-Zeile aufzufrischen. Da das natürlich Zeit kostet, darf die Warteschleife in eben dieser Rasterzeile nicht so groß wie in den anderen sein. Auch die beiden STA $D021 sind nötig, damit das Timing genau stimmt. Selbstverständlich könnte hier auch statt dem zweiten STA $D021 ein STA $D020 stehen: damit wären die Coppers dann auch im Bildschirmrand.

Übrigens: da der verwendete Zeichensatz komplett invers gehalten ist, schimmern die Raster-Bars nur an den freien Stellen des Zeichensatzes durch und befinden sich damit optisch innerhalb der Zeichen.

Wie aber funktioniert das Fein-Scrolling unserer Bars? Dieser Trick ist eigentlich zu einfach: Wir lassen unsere eben besprochene Rastershow-Routine diese Balken immer anzeigen und rotieren anschließend den Farbbereich (»COLTAB«) immer um ein Byte nach links – oder wenn gewünscht auch nach rechts; damit scrollen die Coppers dann nach unten:

```
;-----------------------------------------

SCROLL  LDY COLTAB      ;FARBTABELLE DER
        LDX #$00        ;RASTERFARBEN
SCR1    LDA COLTAB+1,X;UM EIN BYTE
        STA COLTAB,X    ;NACH LINKS
        INX             ;ROTIEREN.DADURCH
        CPX #$40        ;ENTSTEHT EIN
        BNE SCR1        ;FEINES
        STY COLEND-1    ;RASTERSCROLLING
        RTS             ;ENDE
```

Speicherbelegung

$1000 - $2000	Musik (inkl. Player und Init)
$2800 - $3000	Zeichensatz 1
$3000 - $3800	Zeichensatz 2
$4000 - $4450	Demo-Routine (inkl. Texte, Tabellen und Farben.









[4] Bordersprites, Tic Tac, FLYP und Rasterscroller

Das mag zwar unglaublich klingen, dennoch entsteht ein sehr feines Rollen der Bars. Je nachdem wie oft Sie diese Routine hintereinander aufrufen (Achtung: Rasterzeit beachten!) wird das Scrolling schneller oder langsamer. Um Endlos-Scrolling zu ermöglichen, retten wir den jeweils ersten Wert vor dem Überschreiben ins Y-Register und fügen ihn später wieder am Ende der Tabelle an.

Zurück zur Hauptschleife: Hier steht nach diesen Aufregungen ein kleiner Zeichensatzwechsel an. Da wir im oberen Bereich einen großen 4-Byte-Charset verwendet haben, für die drei blinkenden Zeilen aber einen kleinen benötigen, schalten wir einfach das Charset-Register $D018 im VIC um. Für unsere drei blinkenden, bereits im INIT-Part auf den Bild-

Fortsetzung auf Seite 41

Gold Designer – Intromaker mit Niveau

Profi-Intros

Endlich ist das erste selbstproduzierte Programm zusammengeschustert; was jetzt noch fehlt ist ein professionelles Intro. Was unser »Gold-Designer« kann, haut Sie um...

Nach Laden und Starten des Designers begrüßt Sie zunächst eine ausführliche Anleitung, die Sie entweder mit »+« bzw. »—« durchlesen (Abb. 1) oder per <SPACE> abbrechen können. Nach einigen Sekunden Entpackens können Sie dann aber loslegen. Wenn Sie direkt auf Taste <5> drücken, zeigt der Designer ein bereits fertiges Intro (Abb. 2). Mit <SPACE> geht's dann wieder zurück ins Hauptmenü, das sich wie folgt untergliedert:

Taste <1>: Screen-Menü

EDIT SCREEN #01:

Oberen Teil des Bildschirms editieren. Für 4-Byte-Zeichensätze (also 2x2 Characters) müssen Sie zunächst den gewünschten Buchstaben drücken und anschließend denselben Buchstaben bei gedrückter <SHIFT>-Taste. Die zweite Hälfte des Zeichens wird unter den ersten beiden Bytes genauso eingegeben, mit dem kleinen aber feinen Unterschied, daß Sie die REVERSE-Funktion aktivieren müssen. <RETURN> verläßt den Editor wieder.

EDIT SCREEN #02:

Unteren Teil des Bildschirms editieren (maximal drei Zeilen). <RETURN> verläßt den Editor wieder.

TEXT FLASH ON/OFF:

Rhythmisches Blinken des unteren Textbereichs im Intro ein- oder ausschalten.

EDIT SCROLLTEXT:

Hier können Sie den Text eingeben, der später über den Bildschirm scrollt. Insgesamt haben Sie 1000 Zeichen zur Verfügung – also fast eine komplette Bildschirmseite. Achtung: nach Ihrem Text muß unbedingt ein Klammeraffe stehen (»@«), damit der Text immer wieder von neuem beginnt. <CLR> oder <CLR/HOME> arbeiten wie gewohnt. <RETURN> verläßt den Editor.

SCROLLSPEED SLOW/FAST:

Ändern der Scroll-Geschwindigkeit Ihrer Laufschrift.

SCROLLFLASH ON/OFF:

Normalerweise blinkt der Scroller in mehreren Farben. Steht dieser Schalter auf OFF, ist der Text hellgrau.

Taste <2>: Disk-Menü

LOAD MUSIC $1000:

Hier können Sie eine beliebige Musik laden, die allerdings drei Bedingungen erfüllen muß:

1) Speicherbereich zwischen $1000 und maximal $2000.
2) Initialisierung ab $1000.
3) IRQ-Einsprung bei $1003.

Auf der Diskette finden Sie drei bereits fertige Stücke, die Sie nur noch einladen müssen.

LOAD 2X2 CHARSET:

4-Byte-Zeichensätze laden; diese müssen ab $2800 im Speicher vorliegen. Beispiele befinden sich auf der Disk.

LOAD 1X1 CHARSET:

1-Byte-Zeichensätze laden (ab $3000). Drei Beispiele befinden sich auf der Diskette.

LOAD RASTER COL.:

Rasterfarben laden. Die scrollenden Farben im 4-Byte-Zeichensatz lassen sich beliebig verändern. Der Designer erwartet die Werte im Speicher ab $41A0 bis $421F. Für Nichtprogrammierer liegen drei Beispieldateien auf Diskette vor.

SAVE AS A DEMO:

Speichert das komplette Intro als Einzel-File auf Diskette.

SAVE AS AN INTRO:

Erstellte Intros in andere Programm einklinken. Zunächst müssen Sie den Namen des zu behandelten Files angeben. Nach <RETURN> fehlt nur noch der Name des Intros. Das zu linkende File – maximal 134 Blocks – wird jetzt mit dem Intro gekoppelt und komplett auf Disk gespeichert.

INTRO WITH LOADER:

Unter diesem Punkt legt der Gold-Designer Ihr Intro als Einzel-File auf Disk ab. Nach dem Start und Druck auf <SPACE> wird dann das gewünschte File nachgeladen.

Taste <3>: Packer

Achtung: wenn Sie den Packer starten, gehen alle nicht gesicherte Informationen unwiederbringlich verloren, deshalb gibt es auch eine Sicherheitsabfrage. Der Packer ist selbsterklärend. Bei der Frage »CHANGE LOAD-ADRESS?« ein <N> für No eingeben, die Startadresse (»JMP TO:«) auf $4000 setzen und bei »$01:« eine 37 eintragen. Nach Druck auf <RETURN> geht's dann los.

Taste <4>: Directory

Disketteninhaltsverzeichnis anzeigen

Taste <5>: TEST DEMO/INTRO

Hier können Sie sich Ihr Intro anschauen, um eventuell Verbesserungen einfließen zu lassen oder einfach um die geladene Musik oder die Zeichensätze zu überprüfen.

Taste <6>: EXIT PROGRAMM

Gold-Designer verlassen. (pk)

GOLD DESIGNS
PRESENTS
GOLD DESIGNER
WRITTEN BY: SASCHA PETERSEN
COPYRIGHT 64'ER DISC
E YOU TYPE THE NAME OF THE PROG. THAT I

[1] Eine komplette Anleitung ist im Gold-Designer bereits eingebaut

--> ANLEITUNG GOLD-DESIGNER V1 0 <
(SPACE) = GOLD DESIGNER V1 0
(PLUS) = - EINE SEITE VOR -
(MINUS) = - EINE SEITE ZUR -
ANLEITUNG VON: SASCHA PETER EN
MUSIC VON: STEFAN HARTWIG

[2] So könnte ein Vorspann zu Ihren eigenen Programmen bald aussehen

Kurzinfo: Gold-Designer

Programmart: Intro Designer
Laden: LOAD "GOLD-DESIGNER V1",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: eingebauter Packer und Linker
Benötigte Blocks: variabel; min. 77 Blocks
Programmautor: Sascha Petersen

Fraktale auf dem C64

Blick hinter den Spiegel

Kunst aus dem Computer ist nicht nur Großrechnern vorbehalten, auch der C64 kann hier mithalten.

Nur in puncto Bildaufbau ist der C64 natürlich hoffnungslos unterlegen. Doch wenn Sie etwas Zeit haben, können Sie genauso schöne Bilder wie von einem Großrechner erhalten. Obwohl alle zur Berechnung erforderlichen Routinen in Assembler geschrieben sind braucht das Programm für eine Grafik mit maximaler Tiefe bis zu acht Stunden, bis das fertige Bild auf dem Bildschirm steht. Es werden ja für 32 000 Punkte bis zu 250mal quadriert, subtrahiert und verglichen (siehe Textkasten).

Programmbedienung

Nach dem Laden des Programms erscheint das Menü (Abb. 1). Hier lassen sich acht Punkte anwählen:



[1] Menügesteuert lassen sich die Fraktale bearbeiten

(1) Neue Werte

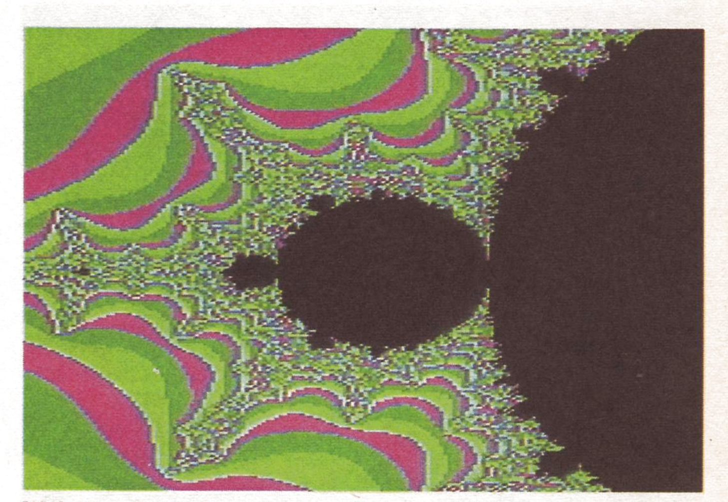
Jetzt können die Grenzen für den Realteil (linker/rechter Rand) und Imaginärteil (oberer/unterer Rand) eingegeben werden. Das Ausgangsbild (Abb. 2) besitzt folgende Parameter:
linker Rand: - 0,7
rechter Rand: 2,1
unterer Rand: - 1
oberer Rand: 1

Kurzinfo: Apfelmännchen

Programmart: Grafikprogramm
Laden: LOAD "APFELMAENNCHEN", 8,1
Starten: nach dem Laden RUN eingeben
Besonderheiten: lädt nach dem Start die zur Berechnung erforderlichen Assembler-Routinen nach
Benötigte Blocks: 23
Programmautor: Gerhard Pehland



[2] Der Urvater der mit diesem Programm berechneten Fraktal



[3] Andere Farbwahl ergibt ein völlig neues Bild



[4] Ein Ausschnitt aus den Ursprungsbild zeigt wieder ähnliche Strukturen

Der nächste vom Programm abgefragte Wert bestimmt die Tiefe. Sie gibt an, nach wie vielen Iterationen (Wiederholungen) der Computer die Berechnung abbricht. Hier sind Werte zwischen 5 und 254 zulässig. Je höher dieser Wert gewählt wird, desto mehr Details treten am Rand der Figur hervor. Ein größerer Wert für die Tiefe verlängert aber auch die nötige Rechenzeit. Zum Ausprobieren der Werte sollten Sie deshalb einen möglichst kleinen Wert eingeben. Man kann dann relativ gut einschätzen, ob das Bild den Vorstellungen entspricht. Ist alles o.k. können Sie das Bild nun mit einer größeren Tiefe berechnen lassen. Was aber bis zur achtfachen Rechenzeit dauern kann.

Zum Schluß der Eingabe werden Sie gefragt, ob das fertige Bild gespeichert werden soll. Das ist besonders praktisch bei Berechnungen, die über Nacht laufen. Sobald der C64 fertig ist, speichert er die Grafik gleich auf Diskette.

Dazu verlangt das Programm dann den File-Namen. An diesen wird automatisch ein .pic angehängt.

Nach Eingabe aller Werte beginnt der Computer mit der Berechnung. Der Bildschirm wird schwarz und Sie können nun den Aufbau des Bildes Zeile für Zeile verfolgen. Ein Abbruch des Programms ist nur noch durch »RUN/STOP-RESTORE« möglich. Jedoch auch eine abgebrochene Berechnung läßt sich speichern. Aus einem Teil dieses Bildes läßt sich nämlich noch ein Ausschnitt berechnen. Dazu muß der nächste Menüpunkt angewählt werden.

(2) Ausschnitt

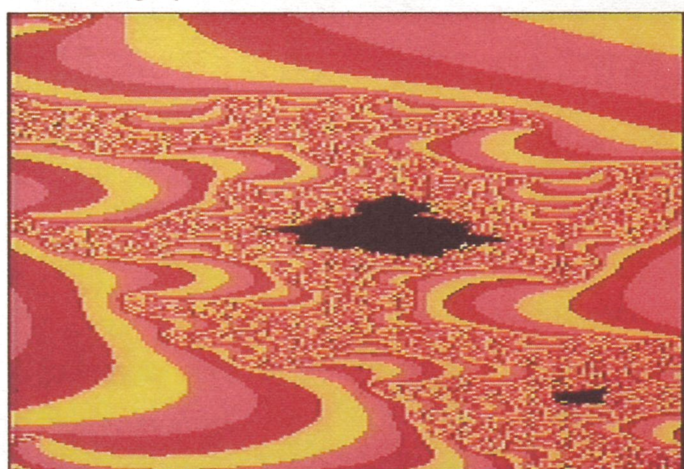
Es erscheint am linken oberen Bildrand ein weißer Winkel, mit dem die linke obere Begrenzung des Ausschnitts bestimmt werden kann. Über die Cursor-Tasten läßt sich die Markierung über den gesamten Bildschirm bewegen. Haben Sie die richtige Position gefunden, bestätigen Sie sie mit der SPACE-Taste. Es erscheint ein zweites Eck unten rechts auf dem Screen. Damit wird nun die rechte untere Begrenzung definiert. Der angewählte Ausschnitt sollte in etwa die Form des Bildschirms haben, da sonst die Grafik stark verzerrt würde. SPACE bestätigt erneut die Einstellung. Nun befinden Sie sich wieder in der Eingabe der Werte. Nach Angabe der Tiefe und dem Namen der Grafik startet die Berechnung.

(3) Bild ansehen

Nach der Berechnung eines Bildes färbt sich der Bildschirm grau. Durch Anwählen dieses Menüpunktes können Sie sich die Grafik wieder auf den Screen holen. Zurück ins Menü gelangen Sie mit dem <←>.

(4) Speichern

Die fertige Grafik kann hiermit auf Diskette geschrieben werden. Dabei wird automatisch die Endung ».pic« an den Namen angehängt. Zusätzlich zu den Grafikdaten werden die Parameter gespeichert.



[5] Die Berechnungstiefe bestimmt die Details der Grafiken



[6] Die interessanten Details befinden sich manchmal am Rand

(5) Laden

Gespeicherte Bilder lassen sich mit diesem Befehl wieder laden. Dabei darf die Endung ».pic« nicht mit eingegeben werden. Nach Beendigung des Ladevorgangs werden die dem Ausschnitt entsprechenden Parameter angezeigt. Mit »Bild ansehen« können Sie sich jetzt die Grafik auf den Screen holen.

(6) Directory

Mit diesem Menüpunkt können alle Bilder auf der Diskette, die mit ».pic« enden, angezeigt werden.

(7) Effekt

Durch zyklisches Vertauschen der Farben entsteht der Eindruck eines bewegten Bildes. Mit den Funktionstasten können auch hier die Farben frei gewählt werden.

(8) Dia-Show

Mit diesem Menüpunkt lassen sich alle auf Disketten gespeicherten Bilder auf Tastendruck in den Computer holen und sofort anzeigen. (jh)

Mathematische Grundlagen

Die fraktalen Gebilde, die der C64 berechnet, beruhen auf einer relativ einfachen mathematischen Formel.

Der C64 berechnet hierbei eine komplexe Zahl. Eine solche Zahl besteht aus einem Real- und einem Imaginärteil. Das Programm stellt die beiden Anteile durch die X- und Y-Koordinate auf dem Bildschirm dar. Jeder Punkt auf dem Screen entspricht den Koordinaten einer komplexen Zahl, die im folgenden »C« genannt wird.

Doch wie entstehen nun diese Grafiken?

Das Programm beginnt mit der komplexen Zahl 0. Davon wird C abgezogen. Das Ergebnis ist eine neue komplexe Zahl, die jetzt quadriert wird. Nun wird wieder C abgezogen, erneut quadriert usw. Es entsteht eine Folge von Zahlen mit einer merkwürdigen Eigenschaft: Bei gewissen Werten von C werden die Elemente dieser Folge rasch kleiner, während sie bei anderen Werten stark ansteigen. Bei anderen Werten pendelt die Folge aber unschlüssig hin und her, bis sie sich schließlich für eine Richtung entscheidet und schnell steigt oder fällt.

Die Farbe eines Punkts auf dem Bildschirm wird bestimmt, indem man den dazugehörigen Wert C in die Folge einsetzt und so lange immer wieder quadriert und abzieht, bis die Elemente der Folge eine bestimmte Grenze überschreiten. Aus der Anzahl der Iterationen ergibt sich nun die Farbe des entsprechenden Punkts. Natürlich muß der Computer die Folge irgendwann einmal abbrechen. Das geschieht zum einen, wenn die Folge das Grenzkriterium überschritten hat oder zum zweiten, wenn die maximale Anzahl der Berechnungen (also die maximale Tiefe) erreicht worden ist.

Werden nun die Berechnungen für alle Punkte des Bildschirms durchgeführt, entstehen die typischen Apfelmännchen.

Hierbei entstehen die interessantesten Muster in der Grenzschicht zwischen dem schwarzen Innenraum und den helleren Außenschichten. Läßt man nämlich dort einen Ausschnitt berechnen, entstehen immer neue, sich selbst ähnliche Muster, die in der Größe variieren. Dabei sollte die Berechnungstiefe sehr hoch angesetzt werden, um dem Computerbild möglichst viele Details zu entlocken. Am Rande des Apfelmännchens sitzen nämlich weitere kleine Apfelmännchen, die dem großen mehr oder weniger ähnlich sind. Hier müssen Sie etwas mit den Werten jonglieren.

(Gerhard Pehland/jh)

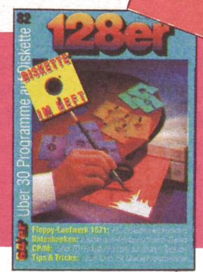
64'er Sonderhefte alle auf einen Blick

Die 64'er Sonderhefte bieten Ihnen umfassende Information in komprimierter Form zu speziellen Themen rund um die Commodore C 64 und C 128. Ausgaben, die eine Diskette enthalten, sind mit einem Diskettensymbol gekennzeichnet.

C 64, C 128, EINSTEIGER



SH 76: C 128
"DiskEtti 128" druckt Diskettenaufkleber/ Mehr Sprites mit "Sprite-Tool"



SH 82: C128
Floppy-Laufwerk 1571 / Datenbanken / CP/M: Diskettenformate für andere Systeme



SH 26: Rund um den C64
Der C64 verständlich für Alle, mit ausführlichen Kursen



SH 36: C 128
Power 128: Directory komfortabel organisieren / Haushaltsbuch: Finanzen im Griff / 3D-Landschaften auf dem Computer



SH 38: Einsteiger
Alles für den leichten Einstieg / Super Malprogramm / Tolles Spiel zum selbermachen / Mehr Spaß am Lernen



SH 50: Starthilfe
Alles für den leichten Einstieg / Heiße Rhythmen mit dem C 64 / Fantastisches Malprogramm



SH 51: C 128
Volle Floppy-Power mit "Rubikon" / Aktienverwaltung mit "Börse 128"



SH 58: 128er
Übersichtliche Buchhaltung zuhause / Professionelle Diagramme



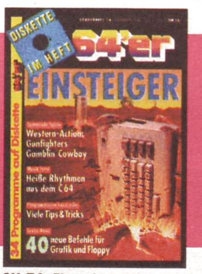
SH 62: Erste Schritte
RAM-Exos: Disketten superschnell geladen/ Exbasic Level II: über 70 neue Befehle/ Raffinessen mit der Tastatur



SH 64: 128er
Anwendungen: USA Journal / Grundlagen: CP/M, das dritte Betriebssystem/ VDC-Grafik: Vorhang auf für hohe Auflösung



SH 70: C 128
Finanzen/ Vereinsverwaltung/ Umwelt/ CP/M-Grundlagen/ Hardware/ Tips&Tricks



SH 74: Einsteiger
Basic 3.5: über 40 neue Befehle und Tastaturfunktionen/ FOBS: Komfortable Benutzeroberfläche/ Tips&Tricks/ Open Access: Dateiverwaltung, Videos, Adressen usw.

PROGRAMMIERSPRACHEN



SH 35: Assembler
Abgeschlossene Kurse für Anfänger und Fortgeschrittene



SH 40: Basic
Basic Schritt für Schritt / Keine Chance für Fehler / Profi-Tools und viele Tips



SH 71: Assembler
Kursus/ Komplettpaket/ Befehlsposter/ Tips&Tricks/ Leserfragen

DTP



SH 39: DTP, Textverarbeitung
Komplettes DTP-Paket zum Abtippen / Super Textsystem / Hochauflösendes Zeichenprogramm

ANWENDUNGEN



SH 78: Anwendungen
Business-Grafik: Statistik zum Anfassen / Raffinierter Soundeditor und 15 Demos / Mit MAS 1.0 zum Einser Abitur

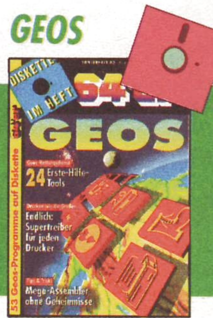


SH 81: Anwendungen
Zeichenprogramm der Superlative: Paint-Mania 64 / Disketten im Griff: Disk-Tool V 6.5 / Der Knopfdruck-Komponist: Maestro 64



SH 68: Anwendungen
Kreuzworträtsel selbstgemacht/ Happy Synth: Super-Synthesizer/ Sir-Compact: Bit-Packer verdichtet Basic- und Assemblerprogramme.

GEOS

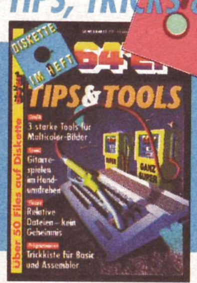


SH 80: GEOS
24 Erste-Hilfe-Tools / Supertreiber für jeden Drucker / Mega-Assembler ohne Geheimnisse

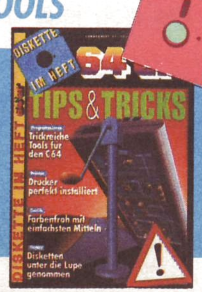


SH 59: GEOS
GeoBasic: Großer Programmierkurs mit vielen Tips & Tricks

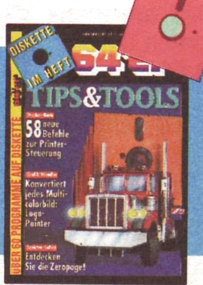
TIPS, TRICKS & TOOLS



SH 77: Tips&Tools
Grafik: Tools für Multicolor-Bilder / Tricks für Basic und Assembler / Floppy: Relative Dateien - kein Geheimnis



SH 57: Tips & Tricks
Trickreiche Tools für den C64 / Drucker perfekt installiert



SH 65: Tips&Tools
Streifzug dirch die Zeropage/ Drucker-Basic: 58 neue Befehle zur Printer-Steuerung/ Multicolorgrafiken konvertieren/ über 60 heiße Tips&Tricks

HARDWARE



SH 25: Floppylaufwerke
Wertvolle Tips und Informationen für Einsteiger und Fortgeschrittene



SH 47: Drucker, Tools
Hardcopies ohne Geheimnisse / Farbige Grafiken auf s/w-Druckern



SH 67: Wetterstation:
Temperatur, Luftdruck und feuchte messen/ DCF-Funkuhr und Echtzeituhr/ Daten konvertieren: vom C64 zum Amiga, Atari ST und PC

64'er Magazin

auf einen Blick

Diese 64'er-Ausgaben bekommen Sie noch bei Markt&Technik für jeweils 7,-DM. Ab Ausgabe 1/92 kostet das Heft 7,80 DM. Die Preise für Sonderhefte und Sammelbox entnehmen Sie bitte dem Bestellcoupon. Tragen Sie Ihre Bestellung im Coupon ein und schicken Sie ihn am besten gleich los, oder rufen Sie einfach unter 089 - 240 132 22 an.

10/91: 100 besten Tips&Tricks / Listing: Fraktal-Programm / C-64-Meßlabor: komfortables Kontrollmodul

11/91: Alles über Diskette & Floppy / Bauanleitung: C 64 steuert Laserstrahl / Sha-Jongg: Topspiel mit Spitzengrafik / Großer Spieleteil

1/92: Viren/ Die neue 64er Floppy/ Neue Produkte-Top-Tests/ Floppy-Kursus für Fortgeschrittene/ Assembler-Corner

2/92: Die Beste Software/ Programm des Monats: The Texter/ Grundlagen- wissen: so programmiert man Packer/ Wettbewerb!

3/92: Tintenstrahler im Vergleich/ der neue Super-Assembler/ Grundlagen: Kopierschutz/ Zum Abtippen: Programme im Heft/ Wettbewerb

4/92: Künstliche Realitäten: Die besten Simulationsprogramme/ C64-Tuning/ Programm des Monats: Vokabeltrainer de Luxe/ Die besten Lernprogramme

5/92: Desktop-Publishing: Alles über DTP, Test DTP-Programme / Scanner: So holt man Bilder in den Computer / Programm des Monats: Top-Adreßverwaltung

6/92: Software auf Knopfdruck: Alles über EPROMs / Datenkonvertierung vom C64 zu Amiga, PC & Atari ST / Programm des Monats: Magazin-Creator de Luxe

7/92: 64ér Jubiläum: Von ´82 bis ´92 / Knallharte Tests: Flüster-Drukker, Geos-Software etc. / Top-Listing: Line V1.0 - Grafik vom Feinsten

8/92: Test: 8 Top-Drucker unter 600 Mark / Hardware: C64 an 12 Volt-Batterie / Daten und Adreßanzeige selbst gebaut / Jede Menge Programme und Tips&Tricks

9/92: Die Besten Joysticks: Newcomer aus England und großer Vergleichstest / Drucker unter 1000 DM auf dem Prüfstand / Assembler für Einsteiger /35 Seiten Tips & Tricks

10/92: Perfekte Filme mit dem C64 / Alle Zeichen- und Malprogramme / Die Kopierschutztricks der Profis / Tests: Drucker-Interfaces, Joystick-Stars (II)

11/92: Heißes Eisen: Softwarerecht / Harddisks, Floppies &Co. /Tests: Musik-Soft-&Hardware, Minijoysticks, Canon BJ 20, 1750 Clone

GRAFIK

SH 75: Grafik
Superfrac: Welt der Fractale / Hi-Eddi: Zeichenprogramm der Spitzenklasse



SH 45: Grafik
Listings mit Pfiff / Alles über Grafik-Programmierung / Erweiterungen für Amica-Paint



SH 63: Grafik
Text und Grafik mischen ohne Flimmern / EGA: Zeichenprogramm der Superlative / 3 professionelle Editoren



SPIELE



Top Spiele 1
Die 111 besten Spiele im Test/ Tips, Tricks und Kniffe zu heißen Games/ Komplettlösung zu "Last Ninja II"/ große Marktübersicht: die aktuellen Superspiele für den C64



SH 30: Spiele für C64 und C128
Spiele zum Abtippen für C64/ C128/ Spieleprogrammierung



SH 37: Spiele
Adventure, Action, Geschicklichkeit / Profihilfen für Spiele / Überblick, Tips zum Spielekauf



SH 42: Spiele
Profispiele selbst gemacht / Adventure, Action, Strategie



SH 49: Spiele
Action, Adventure, Strategie / Sprites selbst erstellen / Virenkiller gegen verseuchte Disketten



SH 52: Abenteuerspiele
Selbstprogrammieren: Von der Idee zum fertigen Spiel / So knacken Sie Adventures



SH 54: Spiele
15 tolle Spiele auf Diskette/ der Sieger unseres Programmierwettbewerbs: Crillion II/ ein Cracker packt aus: ewige Leben bei kommerziellen Spielen





SH 60: Adventures
8 Reisen ins Land der Fantasie - so macht Spannung Spaß



SH 61: Spiele
20 heiße Super Games für Joystick-Akrobaten/ Cheat-Modi und Trainer POKEs zu über 20 Profi-Spielen/ Krieg der Kerne: Grundlagen Spieleprogrammierung

SH 66: Spiele
15 Top-Spiele mit Action und Strategie/ Mondlandung: verblüffend echte Simulation und Super-Grafik/ High-Score-Knacker: Tips&Tricks zu Action-Games

SH 73: Spiele
Action bis Adventure: Zehn Spiele zum Kampf gegen Fabelwesen/ Preview/ Tips&Tricks/ Kurse/ Game Basic/ Mission II/ W.P. Tennis II/ Omnibus GmbH/ Mic's Push´em



SH 79: Spiele
25 superstarke Spiele. Action, Geschicklichkeit, Strategie und die Mini-Parade. Mit diesen Tips&Tricks knacken Sie jedes Spiel.



BESTELLCOUPON

Ich bestelle 64er Sonderhefte

___ Sonderhefte ohne Diskette je 14,- DM ___/___/___/ _____ DM
___ Sonderhefte mit Diskette je 16,- DM ___/___/___/ _____ DM
___ Sonderhefte "128er" je 24,- DM ___/___/___/ _____ DM
___ Sonderheft "Top Spiele 1" 9,80 DM DM

Ich bestelle ____ 64er Magazin Nr. ___/___/___/ _____ DM
je Heft 7,- DM, ab Ausgabe 1/92 je Heft 7,80 DM

Ich bestelle ____ Sammelbox(en) _____ DM
zum Preis von je 14,- DM

Gesamtbetrag _____ DM

Ich bezahle den Gesamtbetrag zzgl. Versandkosten nach Erhalt der Rechnung.

Name, Vorname

Straße, Hausnummer

PLZ, Wohnort

Schicken Sie bitte den ausgefüllten Bestellcoupon an: 64er Leserservice, CSJ, Postfach 140 220, 8000 München 5, Telefon 089/ 240 132 22

Mit 1000 Sachen

Hires-Master – Grafik im Eilzugtempo



Grafik läßt sich beim C 64 nur mit Assembler-Routinen effektvoll realisieren. Wir stellen Ihnen eine der schnellsten Basic-Erweiterungen (wenn nicht die schnellste!) für hochauflösende Grafik vor: Hires-Master.

Grafik-Routinen machen im Prinzip alle dasselbe: Kreise, Linien oder Boxen in Sekundenschnelle auf den Screen bringen. Dennoch unterscheiden sie sich in der Ausführungsgeschwindigkeit oft wie ein Golf von einem Maserati. Hires-Master schlägt alle Rekorde: Mehr als 40 neue Befehle verwandeln den C 64 in eine Grafik-Profimaschine. Im Gegensatz zu vergleichbaren Basic-Erweiterungen wurde die Geschwindigkeit der Grafikroutinen konsequent optimiert, um Bildpunkte auf der Bitmap zu verewigen. So setzt z.B. der LINE-Befehl mehr als 13 000 Pixel pro Sekunde! Bei anderen geometrischen Körpern geht's kaum langsamer. Außerdem verwaltet das Grafiksystem fünf voneinander unabhängige Bildschirme.

Laden Sie die Erweiterung mit:

```
LOAD "HIRES-MASTER",8
```

und starten Sie mit RUN.

Wenn die Einschaltmeldung erscheint, stehen die neuen Befehle für Sie bereit, die sich auch in bereits bestehende Basic-Programme einbauen lassen. Einzige Vorschrift, um Programme mit Hires-Master-Befehlen zu starten: Man muß das neue Grafikbetriebssystem zuvor laden und aktivieren.

Allgemeine Funktionsübersicht

Die bekannten Basic-2.0-Anweisungen gelten weiterhin ohne Einschränkung. Nach einer IF-Anweisung kann sogar THEN entfallen: Die Reaktion auf den IF-Vergleich darf unmittelbar anschließen, z.B.:

```
10 GET T$
20 IF T$="1" PRINT "NR. 1"
30 IF T$="2" PRINT "NR. 2"
40 GOTO 40
```

Hires-Master bietet vier Bitmaps, um Figuren, Kreise, Linien und Rechtecke zu zeichnen. Der fünfte Bildschirm ist temporär, d.h., er wird zum Zwischenspeichern verwendet. Damit erzielt man professionelle Effekte, z.B. verdecktes Zeichnen in Grafik 1, während Nr. 2 auf dem Bildschirm steht. Ist das geänderte Bild fertig, blendet man um (sogar flackerfrei!).

Drei Modi gibt's zum Plazieren von Bildpunkten auf der Hires-Landkarte: Setzen (1), Löschen (0) und Invertieren (2). Die Ausnahme: Der FILL-Befehl arbeitet nur mit dem Modus »Zeichnen« (= 1). Dafür besitzt die Routine eine grandiose Zusatzfunktion: Flächen mit Mustern füllen – was sich speziell bei der Druckausgabe als Riesenvorteil erweist!

Der komfortable TEXT-Befehl positioniert Textfolgen oder einzelne Wörter an beliebiger Stelle im Hires-Bildschirm – sogar in Spiegelschrift. Ebenso lassen sich – wie bei Special Basic oder Simon's Basic – vergrößerte Zeichen ausgeben, wenn man vorher Breite und Höhe mit SIZE bestimmt hat.

Die CIRCLE-Anweisung zeichnet die exaktesten Kreise, die mit dem C 64 zu erzeugen sind – und das quasi in Lichtgeschwindigkeit: Jedes Pixel hat den kleinstmöglichen Abstand zum Idealkreis (bei vielen anderen Grafikprogrammen sind Kreise – unter der Lupe betrachtet – nichts anderes als grob zusammengefügte Vielecke). Diese Genauigkeit ist allerdings schuld daran, daß sich Ellipsen langsamer aufbauen, da der Rechenalgorithmus erheblich mehr Aufwand erfordert.

ARC arbeitet nach dem gleichen Prinzip wie CIRCLE (mit Ellipsenparameter), zeichnet aber Kreis- oder Ellipsenausschnitte. FCIRCLE rundet die Palette der Befehle zum Erzeugen von Rundkörpern ab: Diese Anweisung zeichnet blitzschnell bereits gefüllte Kreise.

Zur Grafikmanipulation gibt's Befehle in Hülle und Fülle: ROLL, SCROLL, DUPLICATE, XMIR und YMIR z.B. verschieben, kopieren oder spiegeln Grafikausschnitte.

Per EFFEKT blendet man zwei Grafikseiten wie im Kino um: Sie können auf 128 verschiedene Spielarten zurückgreifen. Auch das Löschen einer Bitmap erhält mit der Anweisung ECLS professionelles Aussehen.

Wer Teile zweier Bitmaps oder Text- und Grafikbildschirm gleichzeitig anzeigen will, greift auf die WINDOW-Anweisung zurück. Dazu muß man lediglich zwei Bildschirmzeilen bestimmen, bei denen umgeschaltet werden soll, wenn sie der Rasterstrahl erreicht.

Alle Bitmaps (auch die im RAM unter ROM des Betriebsystems ab $E000) lassen sich speichern und laden – unabhängig davon, ob die Grafik eingeschaltet ist oder nicht: Die Bitmap wird weder beim Ein- noch beim Ausschalten gelöscht! Eine separate NMI-Routine (No maskable Interrupt) sorgt dafür, daß nicht einmal die versteckte Grafik im RAM ab $E000 zerstört wird, falls man die Tastenkombination <RUN/STOP RESTORE> benutzt.

Die neuen Basic-Befehle von Hires-Master

Bei der folgenden Erläuterung der neuen Anweisungen tauchen bestimmte Parameterabkürzungen immer wieder auf:
- x: x-Koordinate auf der Bitmap (erlaubte Werte: 0 bis 319),
- y: y-Koordinate, (0 bis 199).

Alle Hires-Master-Anweisungen kann man im Direktmodus oder innerhalb eines Programms verwenden. Für die meisten wurden Befehlsabkürzungen integriert, die Sie in Verbindung mit der SHIFT-Taste eingeben müssen. Man findet Sie in Klammern hinter dem Befehlswortlaut (geSHIFTete Buchstaben sind als Großbuchstaben markiert). Grafikbefehle (LINE, CIRCLE usw.) gelten immer für die mit PAGE eingestellte Bitmap:

HELP (hE)

... zeigt alle Hires-Master-Befehle auf einen Blick.



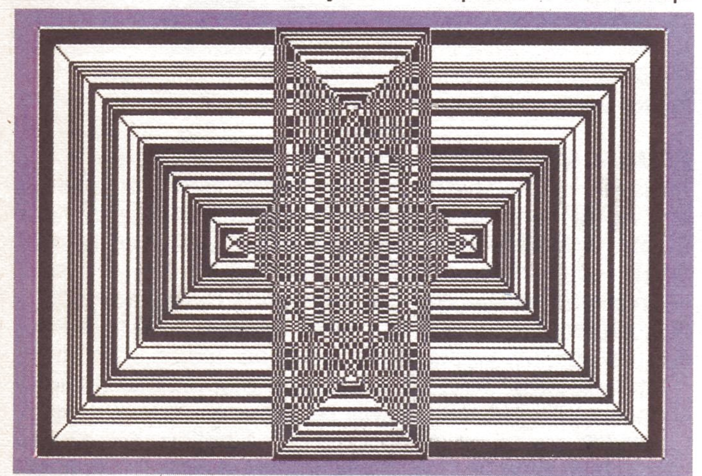
über die Bitmap

LINE x1,y1,x2,y2, (liN)
... zieht einen Strich von x1/y1 bis x2/y2. Ergeben sich aus den Koordinatenangaben horizontale oder vertikal Linien, verzweigt das Programm zu noch schnelleren Unterroutinen.
CIRCLE xm, ym, r (,rx, ry) (cl)
Die Parameter xm/ym bilden stets den Kreismittelpunkt, r bestimmt den Radius (0 bis 255). Werden zwei Radien angegeben (rx/ry), dürfen die Zahlen nicht größer als 128 sein: Es entsteht eine Ellipse.

Mit freundlicher Genehmigung der Firma Fleischmann

INIT (inI)
...versetzt die Basic-Erweiterung in den Einschaltzustand (Vektoren, Farbe, Bitmaps löschen).
CLS
... löscht den aktuellen Grafikbildschirm (alle Bytes = 0). Achtung: Der Befehl nützt beim Textbildschirm nichts! Da hilft nur die Basic-2.0.-Anweisung PRINT CHR$(147).
COLOR pf, hf, (coL)
pf = Vordergrund-, hf = Hintergrundfarbe.
... bestimmt die Farben der Hires-Grafik. Werte zwischen 0 und 15 sind erlaubt.
GRON (gR)
... aktiviert den Grafikbildschirm ab $E000.
GROFF (groF)
... schaltet die Grafik wieder ab. Man befindet sich wieder in dem Bildschirm, der vor dem Einschalten von GRON aktiv war.
MODE m (mO)
... definiert den Zeichenmodus:
- m = 0: gelöschte Pixel,
- m = 1: gesetzter Bildpunkt,
- M = 2: invertieren.

PLOT x,y (pL)
... zeichnet an Position x/y einen Bildpunkt in der Bitmap.



[1] Der BOX-Befehl erzeugt in Sekundenbruchteilen Rechtecke beliebiger Größe

Kurzinfo: Hires-Master

Programmart: Grafik-Basic-Erweiterung
Laden: LOAD "HIRES-MASTER",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: besitzt außergewöhnlich schnelle Grafikroutinen
Benötigte Blocks: 45
Programmautor: Jesko Schwarzer

BLOCK x1, y1, x2, y2 (bL)
... erzeugt ein ausgefülltes Rechteck, dessen linke obere Ecke bei x1/y1 und die rechte untere bei den Koordinaten x2/y2 liegt.
BOX x1, y1, x2, y2 (bO)
... zeichnet - im Gegensatz zu BLOCK - leere Quader (Abb. 1).
FILL x, y (fI)
Mit diesem Befehl kann man z.B. eine leere BOX oder jeden anderen, vollständig umschlossenen Grafikkörper ausfüllen. x/y muß innerhalb der Füllfläche liegen (Abb. 2).

Hires-Master: Zeichenmusterdefinition (Tabelle 1)

Hex	Dez	Bin
00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
0A	10	1010
0B	11	1011
0C	12	1100
0D	13	1101
0E	14	1110
0F	15	1111

Hires-Master: Bitmaps und Farbspeicher (Tabelle 2)

Wert	Bitmap		Farb-RAM	
	Hex	Dez	Hex	Dez
0	$E000	57344	$DC00	56320
1	$2000	8192	$0800	2048
2	$4000	16384	$6000	24576
3	$6000	24576	$5C00	23552
(4)	$8000	32768	$8000	32768
5	$E000	57344	$DC00	56320
6	$E000	57344	$DC00	56320
7	$E000	57344	$DC00	56320

SETMSK fm, m$ (sE)
... definiert Muster für die Füllfläche: bis zu maximal acht lassen sich intern speichern. Die gewünschte Nummer wird mit fm bestimmt (0 bis 7). Achtung: Der FILL-Befehl greift stets auf Muster Nr. 0 zurück (auch wenn Sie höhere Werte für fm angeben)! Will man mit FILL andere Muster auf den Bildschirm bringen, muß Nr. 0 mit anderen ausgetauscht werden (s. unsere Befehlsbeschreibung zu CHAMSK).

So entwirft man Muster für die Stringvariable m$: Es muß aus 16 Zeilen 16 Punkten bestehen (also doppelt so groß wie ein normales Zeichensatzbyte-Raster), jede Zeile besteht aus vier Hexadezimalzahlen. Pro Hexzahl gilt eine Formation von vier Pixeln, die wahlweise gelöscht (=0) oder gesetzt (= 1) sind (Tabelle 1).

Angenommen, Sie entwerfen folgendes 16 x 16-Pixel-Muster:

```
0000 0000 0000 0000 (= 0000)
0000 0011 1100 0000 (= 03C0)
0000 1111 1100 0000 (= 0FC0)
0001 1111 1100 0000 (= 1FC0)
0011 1100 0011 1111 (= 3C3F)
0011 1000 0011 1110 (= 383E)
0111 0000 0011 1100 (= 703C)
0111 0000 0000 0000 (= 7000)
0111 0000 0000 0000 (= 7000)
0111 0000 0011 1100 (= 703C)
0011 1000 0011 1110 (= 383E)
0011 1100 0011 1111 (= 3C3F)
0001 1111 1100 0000 (= 1FC0)
0000 1111 1100 0000 (= 0FC0)
0000 0011 1100 0000 (= 03C0)
0000 0000 0000 0000 (= 0000)
```

Verbinden Sie nun die Hexwerte jeder einzelnen Zeile zur Zeichenkette m$:

```
10 FM = 0
20 M$ = "000003C00FC01FC03C3F383E703C70007000703C
383E3C3F1FC00FC003C00000"
30 SETMSK fm,m$
```

Der String muß stets 64 Zeichen lang sein (pro Zeile vier Hexzahlen = 4 x 16 = 64).

Wollen Sie sich das Muster ansehen, das Sie vorher mit SETMSK definiert haben? (Das Beispiellisting finden Sie auch auf der Diskette zu diesem Sonderheft!)

```
40 GRON: CLS: CIRCLE 160,100,50: FILL 160,100
50 POKE 198,0: WAIT198,1
```

Auf Tastendruck erscheint wieder der Textmodus (Lores-Bildschirm).

CHAMSK ma1, ma2 (chA)

... tauscht interne Füllmuster aus: ma2 ist der Quellbereich, der in den Zielbereich ma1 übertragen wird. So vertauscht man z.B. Muster 0 mit Nr. 7:

```
CHAMSK 0,7
```

STOMSK ma1, ma2 (stoM)

... kopiert Muster Nr. ma1 über Nr. ma2.

ROLL r, x1, y1, x2, y2 (rO)

... scrollt den per x1/y2 (links oben) und x2/y2 (rechts unten) festgelegten Grafikausschnitt um jeweils ein Pixel. r bestimmt die Richtung:

- r = 0: rechts,
- r = 1: links,
- r = 2: hoch,
- r = 3: runter.

Bei ROLL gehen keine Bildpunkte verloren: Falls sie bei der einen Bildschirmgrenze hinausgeschoben werden, kommen sie bei der gegenüberliegenden wieder herein. ROLL sollte man stets in einer FOR-NEXT-Schleife verwenden.

SCROLL r, x1, y1, x2, y2 (sC)

... entspricht der Funktion von ROLL, mit einem Unterschied: Verschwundene Pixel kehren nicht mehr zurück.

TEXT x, y, t$ (tE)

... positioniert den Text t$ an den Bitmap-Koordinaten x/y. Der Textstring darf folgende Steuerzeichen enthalten:

- <CLR/HOME>,
- <HOME>,
- <CRSR aufwärts/abwärts/links/rechts>,
- <RETURN> (= CHR$(13),
- <CTRL N> (= CHR$(14), umschalten zur Kleinschrift,
- CHR$(142) (Großschrift einschalten),
- <RVS ON>,
- <RVS OFF>.

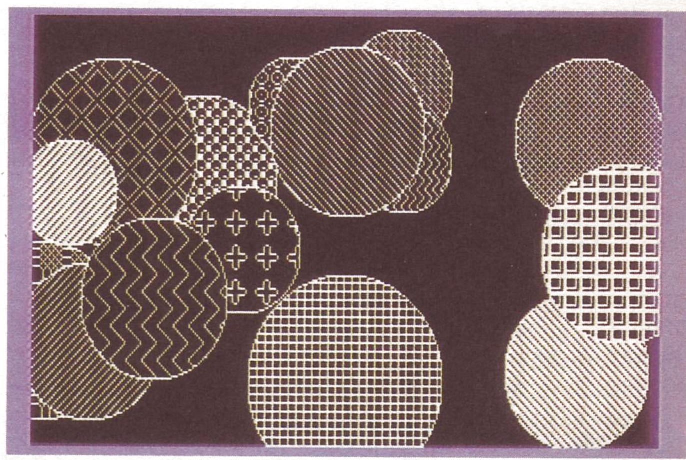
SIZE vx, vy, ax, ay (,d) (siZ)

... bestimmt die Textgröße: per vx in horizontaler, mit vy in vertikaler Richtung. Basis ist die normale Zeichengröße (1 = 8 x 8 Pixel-Raster). Bei 2 sind die Zeichen also doppelt, bei 3 dreimal so groß usw. ax und ay legen den Abstand in Pixeln zwischen den einzelnen Zeichen fest. Damit die Schrift horizontal auf dem Bildschirm erscheint, ist für ay immer 0 zu wählen, jede andere Zahl versetzt den Nachbarbuchstaben um die angegebene Bildpunktzahl nach unten oder nach oben (bei negativen Werten). Dieser Effekt läßt sich mit dem Parameter d (0 bis 3) noch verstärken: Er bestimmt die Anzahl der 90-Grad-Drehungen der Zeichen im Uhrzeigersinn. Auf der Sonderheftdiskette finden Sie das Programmbeispiel »Text« (Abb. 3), das die Befehle SIZE und TEXT verdeutlicht.

ARC xm, ym, rx, ry, sw, ew (aR)

... zeichnet offene Ellipsenausschnitte. Erneut definieren die Koordinaten xm/ym den Kreismittelpunkt, rx/ry die Radien, sw den Start- und ew den Endwinkel im Uhrzeigersinn. 0 Grad ist also bei 12:00 Uhr. Auch das Zeichnen geschieht wie beim Verlauf des Uhrzeigers: Will man z.B. den 45-Grad-Ausschnitt einer Ellipse zeichnen, muß der Endwinkel um 45 Grad größer sein als der Startwinkel.

```
10 GRON: CLS: MODE 1: ARC 160,100,90,30,45,135
1000 POKE 198,0: WAIT 198,1
```



[2] Der Hires-Bildschirm akzeptiert maximal acht verschiedene Muster, um Grafikflächen zu füllen

RAD xm, ym, rx, ry, w (rA)

... zieht die Radiuslinie in einer Ellipse. Die Parameter xm, ym, rx, ry haben die gleiche Bedeutung wie beim ARC-Befehl, w bezeichnet den gewünschten Winkel (0 bis 360 Grad).

Erweitern Sie das Beispielprogramm zu ARC:

```
20 RAD 160,100,90,40,45
30 RAD 160,100,90,40,135
```

Und siehe da: Plötzlich besitzt der zuvor offene Ellipsenausschnitt Begrenzungen (wie bei Tortengrafiken üblich!).

FCIRCLE xm, ym, r (fC)

... zeichnet gefüllte Kreise in der aktuellen Vordergrundfarbe. xm/ym sind die Koordinaten des Kreismittelpunkt, r ist der Radius (0 bis 255).

ECLS verz (eC)

Der Grafikbildschirm wird effektvoll und sanft gelöscht. Die Variable verz bestimmt die Geschwindigkeit (0 = ganz schnell bis 255 = sehr langsam).

REVERS (reV)

... invertiert den aktuellen Grafikbildschirm. Die bei COLOR pf, hf angegebenen Farben werden praktisch vertauscht.

FIGURE x, y, f$ (fiG)

... speichert die relativen Koordinaten eines Grafikkörpers in der Zeichenkette f$, die aber nur aus Zahlen zwischen 1 und 8 bestehen darf:
- 1: ein Pixel nach oben,
- 2: nach rechts,
- 3: nach unten,
- 4: links.

Die Werte 5 bis 8 sind äquivalent zu 1 bis 4, allerdings wird nach jedem Setzen eines Bildpunkts ein Leerpixel übersprungen. Will man z.B. ein Quadrat zeichnen lassen, lautet die Zeichenkette:

```
f$ = "5678"
FIGURE 160,100,f$
```

Ziemlich klein geraten, was? Das Gebilde sieht gerade mal aus wie ein dicker Punkt. Sie könnten jetzt die Zahlen im String vervielfachen, z.B.:

```
f$ = "555555666666777777888888"
```

Doch bevor Sie den Speicher des C64 derart vollmüllen, gibt's einen viel bequemeren Weg:

TURN v, d (tU)

Damit läßt sich der mit FIGURE definierte Grafikkörper ausdehnen und drehen. Der Vergrößerungsfaktor v darf zwischen 0 und 255 liegen, d verträgt Werte von 0 bis 7 und gibt die Anzahl der Drehungen um jeweils 45 Grad an (3 dreht die Figur z.B. um 135 Grad (3 x 45 = 135). Achtung: Damit TURN Einfluß auf FIGURE nehmen kann, muß dieser Befehl vor der FIGURE-Anweisung stehen!



[3] Mit SIZE und TEXT lassen sich Hires-Screens als Dokumente zum Drucker schicken

XMIR x1, y1, x2, y2 (xM)

... spiegelt den durch die Koordinatenpaare x1/y1 und x2/y2 begrenzten Grafikausschnitt an der x-Achse: Er steht auf dem Kopf.

YMIR x1, y1, x2, y2 (yM)

... wie XMIR, berücksichtigt aber die y-Achse: Jetzt erscheint der Grafikteil seitenverkehrt.

PAGE p1, p2, (pA)

... bestimmt den sichtbaren Grafikbildschirm (p2) oder den, in dem verdeckt gezeichnet werden soll (p1). Die Parameter können Werte von 0 bis 7 annehmen. Die Zahlen 0, 5, 6 und 7 bestimmen stets die Bitmap ab $E000 (RAM unterm ROM). Tabelle 2 zeigt die Adressen der Grafik- und Farbspeicher.

Nach dem Start von Hires-Master ist PAGE 7,7 voreingestellt. Bitmap 4 läßt sich nicht einblenden: Dort ist der Zeichensatz abgelegt, außerdem dient dieser Bereich als Zwischenspeicher für den FILL-Befehl. Wer umfangreiche Basic-Programme mit Hires-Master entwickeln will, sollte dann von den Speicherseiten 1 und 2 ebenfalls die Finger lassen.

DUPLICATE x1, y1, x2, y2, x, y (,p1, p2,) (dU)

... kopiert einen Bildausschnitt: x1/y1 (links oben) und x2/y2 (rechts unten) sind die Koordinaten des Bildsegments, das ab der neuen linken oberen Ecke x/y zum zweitenmal im Hires-Screen erscheint. Die PAGE-Parameter p1 und p2 sind optional: Damit läßt sich der Ausschnitt einer Grafikseite in die andere übertragen (p1 = Quell-, p2 = Zielbereich).

CONNECT p1, p2, mo (conN)

... verknüpft einzelne Bitmaps miteinander (z.B. den Inhalt von p1 mit p2), das Ergebnis wird dann in p2 abgelegt. Die Variable mo gibt die Art der Verknüpfung an:
- mo = 0: kopieren,
- mo = 1: OR,
- mo = 2: AND,
- mo = 3: EXOR.

SWAP p1, p2, mo (sW)

Dieser Befehl vertauscht zwei PAGEs (Grafikseiten) und verknüpft sie gleichzeitig: p1 mit p2; p2 wird nach p1 kopiert und das Endergebnis wieder in p2 untergebracht. Der Parameter mo hat dieselbe Bedeutung wie bei CONNECT.

EFFEKT p1, p2, a, verz (eF)

... vertauscht zwei Grafikseiten. Der Parameter a bestimmt die Vertauschungsart und akzeptiert Werte von 0 bis 255, die Variable verz gibt die Wartezeit zwischen dem Kopieren jedes Bytes an (0 bis 255, s. ECLS).

GSAVE p, fi$, ga (gS)

... speichert die Grafik der Bitmap p (s. PAGE) mit dem Dateinamen fi$ auf die Diskette in der Floppy Nr. ga. Dabei wird der Bildschirm vorübergehend abgeschaltet.

GLOAD p, fi$, ga (gL)

... holt das Hires-Bild wieder in den Speicher (ab PAGE p - egal, welche Ladeadresse auf Diskette abgelegt wurde!). Damit lassen sich also beliebige Hires-Grafiken, auch von fremden Zeichenprogrammen oder Basic-Erweiterungen laden. Wichtig: Das Bild darf nicht mehr als maximal 33 Blocks auf Diskette belegen!

WINDOW r1, r2, p1, p2 (wI)

... teilt den Bildschirm in zwei Bereiche: Die Hires-Grafik der Bitmap p1 liegt zwischen den Rasterzeilen r1 und r2; ab r2 bis zum untersten Bildschirmrand sieht man den Grafikbereich aus Bitmap p2. Will man einen der beiden WINDOW-Abschnitte als Textbildschirm (Lores) einsetzen, muß man bei der gewünschten Variablen p1 oder p2 statt der PAGE-Nummer den Wert 128 eintragen.

COPY p(,1) (coP)

... schickt den Grafikbildschirm PAGE p zum seriell angeschlossenen 9-Nadel-Matrix-Drucker (z.B. Star LC-10 oder Epson-kompatible mit Hardware-Interface). Soll der Druck revers erscheinen, muß man zum Parameter p den Wert 128 addieren. Wer eine ganze DIN-A4-Seite bedrucken möchte, muß hinter p noch die Zahl 1 angeben: Der Ausdruck erscheint dann nach rechts gekippt.

Ein Tip: Ändern Sie Zeile 1010 des Listings von »Text«:

```
1010 COPY 7 (oder COPY 7,1)
```

Nach Tastendruck beginnt der Ausdruck des Hires-Bildschirms, den dieses Demoprogramm erzeugt.

CSET x, y, pf (,hf) (cS)

... setzt gezielt Farben an den Koordinaten x/y im Textbildschirm (pf = Vordergrund-, hf = Hintergrundfarbe). X darf zwischen 0 und 39 liegen, Y zwischen 0 bis 24.

OPTION ON/OFF (opT)

... zeigt alle Hires-Master-Befehle revers im Programmlisting (nach Programm voreingestellt). OPTION OFF stellt diese Funktion ab.

OFF (oF)

Damit verläßt man die Basic-Erweiterung. Alle Systemvektoren werden wie bei <RUN/STOP RESTORE> neu initialisiert. Mit SYS 49152 kann man Hires-Master erneut starten.

TEST (x, y) (teS)

... prüft, ob das Pixel am Koordinatenpaar x/y des Hires-Bildschirms gesetzt (= 1) oder gelöscht ist (= 0). Beispiel:

```
PRINT TEST(160,100)
```

Erscheint 1, befindet sich an dieser Stelle ein Grafikpixel.

CTEST (x, y, pp) (cT)

... testet die Farbe des Bildpunkts an den Koordinaten x/y. Ist pp = 0, wird die Vordergrundfarbe überprüft, bei pp = 1 geht's um die Hintergrundfarbe. Das Testergebnis ist immer eine Zahl zwischen 0 und 15 (die üblichen C-64-Farbcodes).

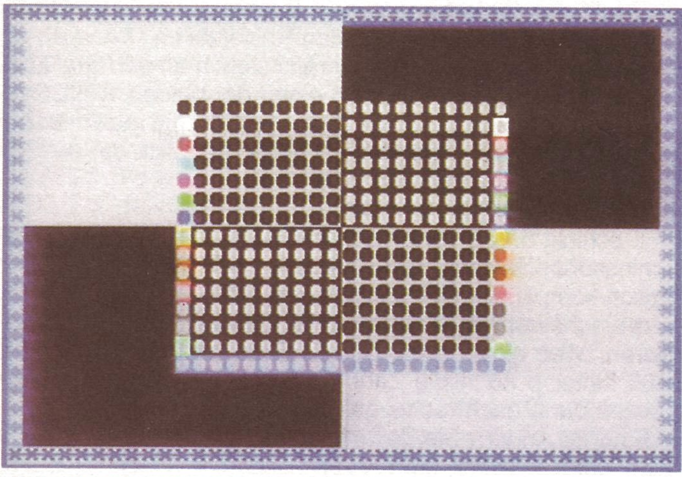
Wir wünschen allen Grafik-Freaks mit dieser rasanten Basic-Erweiterung viel Spaß beim Programmieren! (bl)

Special Basic – mehr als 200 neue Befehle

Grafik im Klartext

Das Basic 2.0 des C 64 ist eine waschechte Provokation, vor allem für Grafikprogrammierer: Ohne unterstützende Assembler-Routinen hat man kaum Chancen, etwas Vernünftiges auf den Bildschirm zu bringen. Doch mit Special Basic beginnt eine neue Ära...

Wer bisher mit PEEK-, POKE-Anweisungen und DATA-Wüsten gearbeitet hat, um per Basic 2.0 nur den Hauch einer Hires-Grafik zu erzeugen, wird begeistert sein: Special Basic besitzt immerhin 43 neue Befehle, die speziell Freunden hochauflösender Grafik oder Sprite-Manipulationen gewidmet sind. Eine Zusammenfassung aller Special-Basic-Anweisungen mit Beispielen finden Sie in unserem Textkasten.



[1] Grafikmanipulationen — auch im Textbildschirm (Lores)



[2] Zwei UFOs – verschollen in den Tiefen des Weltraums

Falls Ihnen Ähnlichkeiten mit der legendären C-64-Basic-Erweiterung »Simon's Basic« auffallen, liegen Sie nicht daneben: Manche Befehle wurden mit gleichem Wortlaut übernommen, andere unterscheiden sich in der Parametereinteilung. Es ist daher kein Problem, Simons-Basic-Programme in Special-Basic-Applikationen umzustricken. Eines muß man allerdings beachten: Beide Erweiterungen verwenden unterschiedliche Token-Tabellen für gleichlautende Anweisungen. Lädt man also Simons-Basic-Programme ins Special-Basic-System, wird z.B. aus COLOUR TURNMOB, aus CENTRE REFLECTY, USE bedeutet DRAW usw. Solche Befehle sind im Simons-Basic-Listing natürlich zu ändern und Parameter anzupassen, um ein lauffähiges Special-Basic-Programm zu bekommen.

Auf der Diskette zu diesem Sonderheft gibt's sieben Demoprogramme, die mit Grafikbefehlen arbeiten.

Laden Sie zunächst die Basic-Erweiterung mit:

```
LOAD "SPECIAL BASIC",8
```

und starten Sie das neue System mit RUN.

Um die Demos in den Speicher zu holen, lassen sich die Anweisungen DLOAD (mit anschließendem RUN) oder CHAIN (Autostart) verwenden, z.B.:

```
DLOAD "GRAFIK-DEMO1"
RUN
```

oder

```
CHAIN "GRAFIK-DEMO1"
```

Grafik-Demo1

... zeigt, wie spielerisch leicht man Hires-Grafiken in Special Basic erzeugen kann: PLOT, DRAW, REC, BOX, CIRCLE usw. SBLOCK und GBLOCK bewegt Bildschirmbereiche in gewünschte Positionen, TEXT bringt Tastaturzeichen in jeder x-beliebigen Größe auf den Monitor. WINDOW erzeugt einen Split-Screen (Text- und Grafikbildschirm geteilt), der sich ideal zur Programmierung von Grafik-Adventures einsetzen läßt. Invertierte Hires-Grafiken bekommt man mit REVERS (wichtig vor allem bei der Druckausgabe, um keine Negativbilder auf Papier zu erhalten).

Mit einer beliebigen Taste schaltet man von einem Hires-Bildschirm zum nächsten. Beim abschließenden Beispiel (TEST-Befehl) verlangt das Programm die Eingabe der x- und y-Koordinaten.

Grafik-Demo2

... kümmert sich um Grafikanweisungen, die man im Textbildschirm (Low Resolution, Lores) aktiviert: Farbige Windows, Screen-Umrandungen, Füllmuster (normal und invertiert, s. Abb. 1), beliebige Positionierung und Scrollen von Textbereichen. Auch hier muß man jedesmal eine Taste drücken, wenn im rechten oberen Bildschirmeck ein blinkender Cursor erscheint.

Kurzinfo: Special Basic

Programmart: Basic-Erweiterung
Laden: LOAD "SPECIAL BASIC",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: bringt über 200 neue Basic-Befehle
Benötigte Blocks: 66
Programmautoren: Michael Störch/Lars George

Special Basic (Speicheraufteilung)

Adreßbereich	Inhalt
$0000 bis $03FF	Zeropage und Pages 1 bis 3
$0400 bis $07FF	Textbildschirm
$0800 bis $7FFF	Basic-RAM und Variablenspeicher
$8000 bis $BFFF	System Special Basic
$C000 bis $C9FF	freies RAM (Assembler-Routinen, Sprites usw.)
$CA00 bis $CAFF	Stack (Schleifen, EXEC-ENDPROC, Flags usw.)
$CB00 bis $CBFF	Funktionstastenbelegung
$CC00 bis $CFFF	Farb-RAM (für Text- und Grafikbildschirm)
$D000 bis $DFFF	temporärer Speicher (TURNMOB, PAINT, GLOBAL/LOCAL)
$E000 bis $FFFF	Hires-Grafik oder neuer Zeichensatz, bzw. Hilfsbildschirme von $F000 bis $F800

Zeichen-Demo

Zunächst wird die wichtigste Funktion zur Zeichensatz-Manipulation vorgestellt: Umlaute und Sonderzeichen definieren (CREATE). In bunter Folge zeigt dann das Programm, wie sich Text auf dem Bildschirm plazieren läßt: in beliebige Richtungen scrollend (TWIST, SDOWN, SRIGHT, SLEFT, SUP), horizontal bzw. vertikal oder gespiegelt (MIRX und MIRY). Gut gelungen: Die Simulation eines Kilometerzählers, der mit den Anweisungen CHAR, SCROLL, CREATE und CHAR$ das langsame Umschalten der Ziffern zum nächsthöheren Wert zeigt.

Sprite-Demo1

... bringt einen kleinen Zeichentrickfilm mit Sprites im Lores-Bildschirm. Die entsprechenden Listingzeilen demonstrieren, wie man Hires- und Multicolor-Sprites (Mann, Fahne, Vogel und UFO) mit CREATE, MOB, DEFMOB, MOBSET und BLOR generiert. MOVE steuert die Bewegung der flinken Kobolde.

Sprite-Demo2

... zeigt zwei UFOs, die durch versteckte Spiralnebel der Bildschirm-Galaxis streifen (Abb. 2). Die Sprite-Bewegung wird ebenfalls durch MOVE in Verbindung mit dem Zufallsgenerator RND aktiviert. Da der Hires-Bildschirm aktiv ist, muß man bei CREATE (Sprites) den Parameter »2« eintragen. Das Demo-Progamm läßt sich mit weiteren Special-Basic-Befehlen jederzeit zu einem Spiel ausbauen - benutzen Sie die Befehle BUMP, um Sprite-Kollisionen festzustellen, zur Joystick-Abfrage die Anweisung JOY(2) usw.

Sternenhimmel und Graphiccharts

... sind umfangreichere Special-Basic-Grafikanwendungen (s. separate Beschreibung).

Special Basic benutzt den Speicherbereich ab $8000 (32768). Unsere Tabelle zeigt die Speicheraufteilung des neuen Basic-Systems, um eigene Maschinensprache-Routinen oder Programmergänzungen einzubauen. Das System verwendet die Modulkennung CBM80 ab $8000, um sich resetfest im Speicher einzunisten. Um wieder ins Basic 2.0 des C64 zurückzukehren, gibt's den Befehl GO 64 oder die Anweisung: POKE 32773,0: SYS 64738. (bl)

Special Basic (Befehlsübersicht)

Special Basic gehört zu den mächtigsten Basic-Erweiterungen für den C64. Das Programm ähnelt Simon's Basic, besitzt aber weit mehr Befehle. Viele Routinen erinnern in Syntax und Funktion z.B. ans Super-Basic V des Acorn Archimedes: PROC – END PROC, CASE OF – WHEN – OTHERWISE – ENDCASE, CHAIN, SWAP, LOMEM, HIMEM. LOCAL, GLOBAL usw.

Die markantesten Programmfunktionen widmen sich der hochauflösenden Grafik und den Sprites: 43 neue Befehle wurden dafür reserviert! Kleiner Wermutstropfen: Special Basic unterstützt zwar farbige Hires-Grafiken, aber nicht den Multicolormodus. Normale Basic-2.0-Programme laufen ohne Einschränkung, sofern diese keine separaten Assembler-Routinen im Speicherbereich ab $8000 (32768) verwenden oder länger als ca. 100 Blocks sind. Alle Basic-2.0-Anweisungen behalten ihre Gültigkeit.

Zur Beschreibung der neuen Basic-Befehle: Parametervariablen in eckigen Klammern sind optional. Sie weisen entweder auf mehrere Variationen der Befehlseingabe hin oder dürfen vollständig entfallen.

I. Programmierhilfen

AUTO zn,s

Parameter: zn = erste Zeilennummer, s = Schrittweite

Automatische Zeilennumerierung bei Eingabe von Basic-Programmzeilen. Auto-Modus verlassen: per <RETURN> unmittelbar hinter der Zeilennummer.

RENUM [zn,s]

Parameter: zn = erste neue Zeilennummer, s = Schrittweite

Die Basic-Programmzeilen werden neu numeriert. Sprungverweise bei GOTO, GOSUB, RUN, THEN, LIST und RESET passen sich automatisch an. Sprünge auf nicht existierende Zeilennummern werden gemeldet. RENUM ohne Parameter entspricht RENUM 10,10.

DELETE [–zn/zn–/zn–zn]

Parameter: zn = Zeilennummer

...löscht entweder alle Zeilen bis einschließlich zn, alle Zeilen ab zn oder den Bereich von zn bis zn.

OLD

Parameter: keine

...reaktiviert ein durch KILL, NEW oder per Resetknopf gelöschtes Basic-Programm.

TRACE [n]

Parameter: n = Geschwindigkeit (0 = schnell, 255 = sehr langsam)

Programmablauf testen. Die aktuelle Zeilennummer erscheint in der linken oberen Bildschirmecke. Mit dem Wert n stellen Sie die Ablaufgeschwindigkeit ein.

Modus abschalten: TRACE ohne Parameter eingeben!

STRACE

Parameter: keine

Damit lassen sich die einzelnen Programmschritte in den obersten beiden Bildschirmzeilen verfolgen (Single-Step-Trace-Modus). Das Testprogramm stoppt nach jedem Basic-Befehl und macht erst nach Tastendruck weiter. <N> normalisiert den Programmablauf, <S> aktiviert erneut den Einzelschrittmodus.

Abschalten: TRACE ohne Parameter.

MONITOR

Parameter: keine

...aktiviert einen separat geladenen Maschinensprache-Monitor (z.B. SMON), dessen Startadresse mit SET übergeben wurde. Achtung: Der Programmbereich des Monitors darf Adresse $8000 (32768) nicht überschreiten (sonst wird »Special Basic« zerstört!).

SET sa

...definiert die Startadresse des MONITOR-Befehls, die unter $8000 liegen muß. Ebenso sollten Sie daran denken, die Obergrenze des freien Basic-RAMs unter die erste Adresse des Speicherbereichs für den Maschinensprache-Monitor zu setzen, sonst wird dieser von Stringvariablen Ihres Basic-Programms überschrieben.

Beispiel: Sie möchten einen Monitor ab $6000 (24576) in den Computerspeicher holen, dann muß die RAM-Obergrenze heruntergesetzt und die Adresse eingestellt werden:

```
POKE 56,96: SET 24576
```

DUMP

Parameter: keine

...bringt eine Liste aller benutzten Variablen und deren aktuelle Werte. Per <SHIFT> hält man die Listenausgabe auf dem Bildschirm an. DUMP zeigt nur einfache Variablen, keine indizierten (z.B. A$(I)).

MATRIX

Parameter: keine

...zeigt alle Indexvariablen (Arrays) mit aktuellen Inhalten. Die Bildschirmausgabe wird ebenfalls per <SHIFT> angehalten.

FIND zk/var

Parameter: zk = Zeichenkette oder Basic-Befehl, var = numerische oder Stringvarible

...sucht einzelne Zeichen, Strings oder Befehle im Programmtext. Zeilennummern, in denen der Begriff vorkommt, werden ausgegeben.

Beispiele:

```
FIND PRINT
FIND "TEXT"
FIND A$
FIND B%
```

KEY n,a$

Parameter: n = Funktionstastennummer (1 bis 8), a$ = Belegungstext (maximal 31 Zeichen)

Damit können Sie <F1> bis <F8> mit beliebigem Text (z.B. Special-Basic-Befehlen) belegen. Hängt man den <←> an, wird der Befehl unmittelbar ausgeführt (wie nach Tipp auf die RETURN-Taste).

Beispiel:

```
KEY 7, "LIST"
```

Special Basic bietet nach dem Start diese Funktionstasten-Belegung:

- <F1>: COMMANDS,
- <F2>: SHOW,
- <F3>: RUN,
- <F4>: CHAIN,
- <F5>: LIST,
- <F6>: MEMORY,
- <F7>: HELP,
- <F8>: DIR

Geänderte Texte lassen sich als Datei unter beliebigem Namen speichern, z.B.:

```
BSAVE "F-TASTEN", $CB00, $CC00
```

und mit der Anweisung BLOAD "F-TASTEN" wieder laden.

SHOW

Parameter: keine

...bringt die aktuelle Liste der Funktionstasten-Belegung auf den Bildschirm.

OFF

Parameter: keine

Fortsetzung auf Seite 22

So finden Sie die Programme auf der Diskette

DISKETTE SEITE 1

```
Ø "64'er sonderh.87" s1 2a
28  "disklader"         prg  S. 21
Ø   "┌────────────┐"    usr
Ø   "| grafik      |"   usr
Ø   "| erweiterung |"   usr
Ø   "└────────────┘"    usr
66  "special basic"     prg  S. 16
14  "grafik-demo1"      prg
8   "grafik-demo2"      prg
14  "zeichen-demo"      prg
2Ø  "sprite-demo1"      prg
8   "sprite-demo2"      prg
Ø   "----------------"  usr
46  "graphiccharts"     prg  S. 32
2   "Umsatz 1992.dat"   seq
Ø   "----------------"  usr
59  "sternenhimmel"     prg  S. 31
Ø   "----------------"  usr

45  "hires-master"      prg  S. 12
18  "hires-demo"        prg
1   "setmsk"            prg
3   "text"              prg
2   "text2"             prg
Ø   "┌────────────┐"    usr
Ø   "|   grafik    |"   usr
Ø   "|   tools     |"   usr
Ø   "└────────────┘"    usr
3   "picture tool 1.Ø"  prg  S. 44
1Ø  "graph finder lo"   prg
9   "graph finder hi"   prg
12  "graph formatter"   prg
1Ø  "char finder lo"    prg
1Ø  "char finder hi"    prg
7   "char converter"    prg

Ø   "----------------"  usr
26  "gld v2.Ø/by amok"  prg  S. 45
9   "charset         "  prg
5   "object screen   "  prg
51  "level           "  prg
Ø   "┌────────────┐"    usr
Ø   "|  fractals   |"   usr
Ø   "└────────────┘"    usr
58  "apfelmann"         prg  S. 8
23  "apfelmaennchen"    prg
4   "apfelroutinen"     prg
32  "apfelmann.pic"     prg
32  "apfelmaenn.pic"    prg
Ø   "┌────────────┐"    usr
Ø   "|  diskette   |"   usr
Ø   "| beidseitig  |"   usr
Ø   "|  bespielt   |"   usr
Ø   "└────────────┘"    usr
```

DISKETTE SEITE 2

```
Ø "64'er sonderh.87" s2 2a
Ø   "┌────────────┐"    usr
Ø   "|   intros    |"   usr
Ø   "└────────────┘"    usr
77  "gold-designer v1"  prg  S. 7
17  "tune #Ø1/g.d."     prg
12  "tune #Ø2/g.d."     prg
12  "tune #Ø3/g.d."     prg
1   "raster col.#Ø1"    prg
1   "raster col.#Ø2"    prg
1   "raster col.#Ø3"    prg
9   "1x1 char #Ø1"      prg
9   "1x1 char #Ø2"      prg
9   "1x1 char #Ø3"      prg
9   "2x2 char #Ø1"      prg
9   "2x2 char #Ø2"      prg
9   "2x2 char #Ø3"      prg
Ø   "----------------"  usr
25  "demo-source /ta"   prg  S. 4
4Ø  "demo-source /seq"  seq
25  "demo      /ready"  prg
Ø   "┌────────────┐"    usr
Ø   "|   sprites   |"   usr
Ø   "└────────────┘"    usr
23  "sprite-eddi 864"   prg  S. 46
5   "spr-ripper $1ØØØ"  prg
Ø   "----------------"  usr

2   "alle 8 sprites"    prg  S. 48
2   "demo-sprites"      prg
4   "anima+overlay"     prg
1   "ass-sprite $1ØØØ"  prg
Ø   "┌──────────────┐"  usr
Ø   "| tips & tricks |" usr
Ø   "└──────────────┘"  usr
3   "calcul-ace"        prg  S. 34
27  "graphix"           prg
39  "bas"               prg
2   "allsprites"        prg
6   "irq.obj"           prg
Ø   "----------------"  usr
6   "super-cursor"      prg  S. 34
Ø   "----------------"  usr
11  "smile"             prg  S. 34
Ø   "----------------"  usr
12  "koalamodify"       prg  S. 35
4Ø  "▒pic p wern     "  prg
4Ø  "▒pic c koala    "  prg
Ø   "----------------"  usr
6   "slide-konvert"     prg  S. 36
16  "▒Ø8 lucky luke"    prg
Ø   "----------------"  usr
17  "gfxrip $1ØØØ"      prg  S. 36
2   "probesprite"       prg
Ø   "----------------"  usr

1   "hi-eddi/simon"     prg  S. 35
1   "he/sb"             prg
Ø   "----------------"  usr
2   "special-loader"    prg  S. 35
1   "special-saver"     prg
Ø   "----------------"  usr
6   "stamp-packer v1"   prg  S. 36
5   "stamp-maker v2"    prg
1   "stamp-repack"      prg
4   "pack-part 2"       prg
1   "zero-kill"         prg
13  "b skat"            prg
Ø   "----------------"  usr
3   "random-demo"       prg  S. 33
1   "random copy"       prg
1   "bitmap"            prg
32  "globus"            prg
32  "stimmung"          prg
Ø   "----------------"  usr
2   "scr.mover-demo"    prg  S. 42
1   "scr.mover"         prg
Ø   "----------------"  usr
14  "iff-converter"     prg  S. 43
4   "con"               prg
4   "ladercon"          prg
Ø   "┌────────────┐"    usr
Ø   "|    ende     |"   usr
Ø   "└────────────┘"    usr
```

WICHTIGE HINWEISE

zur beiliegenden Diskette:

Aus den Erfahrungen der bisherigen Sonderhefte mit Diskette, wollen wir ein paar Tips an Sie weitergeben:

1

Bevor Sie mit den Programmen auf der Diskette arbeiten, sollten Sie unbedingt eine Sicherheitskopie der Diskette anlegen. Verwenden Sie dazu ein beliebiges Kopierprogramm, das eine komplette Diskettenseite dupliziert.

2

Auf der Originaldiskette ist wegen der umfangreichen Programme nur wenig Speicherplatz frei. Dies führt bei den Anwendungen, die Daten auf die Diskette speichern, zu Speicherplatzproblemen. Kopieren Sie daher das Programm, mit dem Sie arbeiten wollen, mit einem File-Copy-Programm auf eine leere, formatierte Diskette und nutzen Sie diese als Arbeitsdiskette.

3

Die Rückseite der Originaldiskette ist schreibgeschützt. Wenn Sie auf dieser Seite speichern wollen, müssen Sie vorher mit einem Diskettenlocher eine Kerbe an der linken oberen Seite der Diskette anbringen, um den Schreibschutz zu entfernen. Probleme lassen sich von vornherein vermeiden, wenn Sie die Hinweise unter Punkt 2 beachten.

ALLE PROGRAMME
aus diesem Heft



HIER



Markt&Technik
Verlag Aktiengesellschaft

Diskette zum
Sonderheft
Nr.__________

Diese Diskettentasche besteht aus chlorfrei gebleichtem Papier

Chefredakteur: Georg Klinge (gk) – verantwortlich für den redaktionellen Teil
Stellv. Chefredakteur: Arnd Wängler (aw)
Textchef: Jens Maasberg
Redaktion: Harald Beiler (bl), Heinz Behling (hb), Peter Klein (pk), Jörn-Erik Burkert (lb), Hans-Jürgen Humbert (jh)
Producer: Andrea Pfliegensdörfer
Redaktionsassistenz: Birgit Misera, Helga Weber
Mitarbeiter dieser Ausgabe: Herbert Großer

So erreichen Sie die Redaktion:
Tel. 089/46 13-202, Telefax: 089/46 13-5001, Btx: 64 064

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, so muß das angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in den von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträgern. Mit Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß die Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Layout: Dagmar Portugall, Uschi Böcker
Bildredaktion: Roland Müller, Tina Steiner (Fotografie)
Titelgestaltung und -grafik: Erich Schulze

Anzeigenleitung: Peter Kusterer
Anzeigenverwaltung und Disposition: Christopher Mark (421)

Anzeigen-Auslandsvertretung:
Großbritannien und Irland: Smyth International, Telefon 0044/81340-5058, Telefax 0044/81341-9602
Niederlande und Belgien: Insight Media, Telefon 0031/2153 12042, Telefax 0031/2153 10572
Italien: Medias International, Telefon 0039/31 75 14 94, Telefax 0039/31 75 14 82
USA und Kanada: M & T International Marketing, Telefon 001/415358-9500, Telefax 001/415358-9739
Japan: Media Sales Japan, Telefon 0081/33504-1925, Telefax 0081/33595-1709
Taiwan: Acer TWP Corporation, Telefon 00886-2-7136959 Telefax 00886-2-7 151950
Korea: Young Media Inc., Telefon 0082-2-7 564819, Telefax 0082-2-7575789
Israel: Baruch Schaefer, Telefon 00972-3-5562256, Telefax 00972-3-5566944
International Business Manager: Stefan Grajer 089/46 13-638

So erreichen Sie die Anzeigenabteilung:
Tel. 089/46 13-962, Telefax: 089/46 13-791

Leiter Vertriebsmarketing: Benno Gaab

Vertrieb Handel: MZV, Moderner Zeitschriften Vertrieb GmbH & Co. KG, Breslauer Straße 5, Postfach 11 23, 8057 Eching, Tel. 089/31 9006-0

Verkaufspreis: Das Einzelheft kostet DM 16,-

Leitung Technik: Wolfgang Meyer (887)

Druck: SOV. Graphische Betriebe, Laubanger 23, 8600 Bamberg

Sonderdruck-Dienst: Alle in dieser Ausgabe erschienenen Beiträge können für Werbezwecke in Form von Sonderdrucken hergestellt werden. Anfragen an Klaus Buck, Tel. 089/46 13-180, Telefax 089/46 13-232

Vorstand: Carl-Franz von Quadt (Vorsitzender), Dr. Rainer Doll, Dieter Streit

Verlagsleiter: Wolfram Höfler
Operation Manager: Michael Koeppe

Direktor Zeitschriften: Michael M. Pauly

Anschrift des Verlags: Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/46 13-0, Telex 522052, Telefax 089/46 13-100

ISSN 0931-8933

Die Zeitschrift wird mit chlorfreiem Papier hergestellt.

Copyright-Erklärung

Name: ....

Anschrift: ....

Datum: ....

Computertyp: ....

Benötigte Erweiterung/Peripherie: ....

Datenträger: Kassette/Diskette ....

Programmart: ....

Ich habe das 18. Lebensjahr bereits vollendet

...., den ....

....
(Unterschrift)

Wir geben diese Erklärung für unser minderjähriges Kind als dessen gesetzliche Vertreter ab.

...., den ....

....

....

Bankverbindung:

Bank/Postgiroamt: ....

Bankleitzahl: ....

Konto-Nummer: ....

Inhaber des Kontos: ....

Das Programm/die Bauanleitung: ....

das/die ich der Redaktion der Zeitschrift 64'er übersandt habe, habe ich selbst erarbeitet und nicht, auch nicht teilweise, anderen Veröffentlichungen entnommen. Das Programm/die Bauanleitung ist daher frei von Rechten anderer und liegt zur Zeit keinem anderen Verlag zur Veröffentlichung vor. Ich bin damit einverstanden, daß die Markt & Technik Verlag AG das Programm/die Bauanleitung in ihren Zeitschriften oder ihren herausgegebenen Büchern abdruckt und das Programm/die Bauanleitung vervielfältigt, wie beispielsweise durch Herstellung von Disketten, auf denen das Programm gespeichert ist, oder daß sie Geräte und Bauelemente nach der Bauanleitung herstellen läßt und vertreibt bzw. durch Dritte vertreiben läßt.

Ich erhalte, wenn die Markt & Technik Verlag AG das Programm/die Bauanleitung druckt oder sonst verwertet, ein Pauschalhonorar.

Disklader – Programme laden mit Komfort

Diskettenoberfläche de Luxe

Keine umständlichen Ladeanweisungen und ein übersichtliches Inhaltsverzeichnis der Diskette auf dem Bildschirm. Unser »Disklader« erfüllt auch gehobene Ansprüche.

Entwicklungshelfer sind gefragt, denn noch immer sind einige Arbeitsschritte nötig, um beim C64 ein Inhaltsverzeichnis von der Diskette zu erhalten. Außerdem erschweren manche Unterdateien zu einem Programm die Übersicht im »Directory«. Genau hierfür finden Sie einen »Feuerwehrmann« auf der ersten Seite der beiliegenden Diskette - den »Disklader«. Er generiert eine Benutzeroberfläche für Ihren C64. Darin sind Funktionen integriert, wie:

- Anwahl einzelner Programme (mit jeweiliger Kurzbeschreibung),
- automatisches Laden und Starten von Diskette oder
- Erkennung der richtigen Diskette bzw. Diskettenseite.

Da sich der Disklader an erster Stelle auf der Diskette zum Sonderheft befindet, genügt es, zum Laden einzugeben:

```
LOAD":*",8
```

Nach der Bestätigung mit <RETURN> dauert es ca. 15 s, bis die Datei im Speicher ist. Sie starten mit RUN und <RETURN>. Anschließend wird das File entpackt (ca. 2 s) und es erscheint die Benutzeroberfläche des »Disklader« (s. Abbildung). In der rechten unteren Bildschirmhälfte sehen Sie weiß umrandet den Namen des ausgewählten Programms. Die unterste Bildschirmzeile ist die dazugehörige Kurzerklärung. Zusätzlich finden Sie in der rechten unteren Bildschirmhälfte den Text »Seite 1 auf Disk« oder »Seite 2 auf Disk«. Da Sie die Inhaltsverzeichnisse beider Seiten (ohne die Disk zu wenden) durchblättern können, finden Sie hier den Hinweis, auf welcher



Kurzinfo: Disklader

Programmart: Hilfsprogramm zum Laden der Programme auf der beiliegenden Diskette
Laden: LOAD":*",8
Starten: nach dem Laden mit RUN
Steuerung: Tastatur
Programmautor: H. Großer

Diskettenseite sich das gewählte Programm befindet. Durch Tastendruck <CRSR aufwärts> bzw. <CRSR abwärts> wählen Sie das nächste oder vorherige Programm. Sie blättern quasi durch den Inhalt der Programme. <HOME> bringt Sie zum ersten Eintrag des Inhaltsverzeichnisses. Selbstverständlich sind nur die Programme verzeichnet, die sich eigenständig laden oder starten lassen.

<RETURN> führt Sie in den Ladeteil. Ist kein Diskettenfehler aufgetreten, erscheint kurzzeitig »00,OK,00,00« am Bildschirm. Eventuelle Fehleranzeigen bleiben sichtbar am Bildschirm (z.B. »21,READ ERROR,18,00«= Drive not ready). Sie lassen sich durch einen beliebigen Tastendruck wieder löschen. Schlagen Sie bitte vorher im Handbuch Ihrer Floppy nach und beseitigen Sie den Fehler. Eine andere Art der Fehlermeldung wird durch einen blinkenden Text dargestellt (z.B. »Bitte Disk wenden« oder »Falsche Diskette«). Sind Fehler ausgeblieben, lädt der Disklader das von Ihnen gewählte Programm von der Diskette und startet es. Ladefehler, die in dieser Phase auftreten, werden nicht mehr berücksichtigt: Der Disklader wird vom neuen Programm einfach überschrieben. Sonst könnten wir nur Programme veröffentlichen, die mit der Benutzeroberfläche zusammenarbeiten. Bei vielen Spielen, Tricks oder Tools ist dies aber nicht der Fall.

Für Sie bedeutet dies, nach jedem Starten eines Programms den »Disklader« erneut zu laden. Wer die Benutzeroberfläche verlassen will, gibt <RUN/STOP> ein. Sie befinden sich dann im normalen »Basic« des C64. Für einen Neustart befehlen Sie `SYS 12032` und bestätigen mit <RETURN>. Dieser Neustart funktioniert auch nach einem Reset, d.h. wenn Sie durch den entsprechenden Taster einen Hardware-Reset ausgelöst haben. Allerdings sollten Sie zwischenzeitlich kein Programm geladen haben, da dies den verwendeten Speicherbereich überschreiben könnte. Laden Sie in diesem Falle den Disklader neu.

Wir haben bei der Programmierung größten Wert auf Kompatibilität mit den unterschiedlichsten Betriebssystemerweiterungen gelegt. Lediglich bei der Gerätekonfiguration C128 mit RAM-Erweiterung und zweiter Diskettenstation sollten Sie die externe Floppy ausschalten. (gr)

Fortsetzung von Seite 17

...löscht die F-Tastentexte. Die Werte entsprechen nun der Originalbelegung des Basic 2.0.

DELAY n

Parameter: n = Verzögerungswert (0 = schnell, 255 = langsam)

...verzögert die Ausgabe eines Basic-Listings auf dem Bildschirm. Voraussetzung: Sie müssen gleichzeitig die Commodore-Taste drücken! <SHIFT> hält die Listingausgabe an, bis man die Taste wieder losläßt.

PAGE [n])) #

Parameter: n = Zeilenanzahl (1 bis 25)

...stoppt nach der Eingabe von LIST in Abschnitten von n Zeilen. Nach einem Tastendruck macht die Routine mit dem nächsten Bereich weiter.

Abschalten: PAGE ohne Parameter.

MEMORY

Parameter: keine

...zeigt die momentan gültige Speicheraufteilung: Programm, Variablen, dimensionierte Felder (Arrays), freier Basic-Speicher, Stringspeicher-Belegung.

II. Programmstrukturen

Die Möglichkeiten der IF-THEN-Anweisung wurden komfortabel ausgebaut:

IF – THEN

```
10 IF A$ = "JA" THEN 700
```

...die bekannte IF-THEN-Verzweigung des Basic 2.0.

IF – THEN – ELSE

```
10 IF A$ = "JA" THEN 700: ELSE END
```

...ist die Bedingung erfüllt, führt das Programm den JA-Zweig aus, sonst (ELSE) wird die NEIN-Reaktion aktiviert.

IF – THEN – BEGIN – BEND

Der Unterschied zur normalen IF-THEN-Abfrage: Die JA-Verzweigung kann sich zwischen BEGIN und BEND über mehrere Zeilen erstrecken:

```
10 IF A$="JA" THEN BEGIN: PRINT "O.K."
20 POKE 120,0
30 PRINT "FERTIG"
40 BEND
```

IF – THEN – BEGIN – BELSE – BEND

...entspricht der vorhergehenden Anweisung, allerdings kann nun der JA- und NEIN-Zweig mehrere Zeilen lang sein:

```
10 IF A$ = "JA" THEN BEGIN: PRINT "OK"
20 COLOR 7,14,14
30 BELSE
40 PRINT "TSCHUESS": END
50 BEND
```

REPEAT – UNTIL x

Parameter: X = Bedingung

Die Schleife wird so lange ausgeführt, bis die UNTIL-Bedingung erfüllt ist.

```
10 REPEAT
20 A = A+1: PRINT A
30 UNTIL A = 200
```

Maximal 20 Schleifen dürfen ineinander verschachtelt werden.

LOOP – EXIT – END LOOP

Diese Schleife kann man auf zwei Arten verlassen: Mit EXIT auf jeden Fall, per EXIT IF nur dann, wenn die dahinter angegebene Bedingung erfüllt ist.

```
10 LOOP
20 GET A$
[30 IF A$ = "J" THEN EXIT] oder
[30 EXIT IF A$ = "J"]
40 PRINT A$
50 END LOOP
```

Auch dieser Befehl verträgt nur maximal 20 verschachtelte Schleifen.

PROC name(var1, var2)

Parameter: name = Bezeichnung der Prozedur, var1/var2 = diverse Variablennamen

...definiert Name und Beginn einer Programmprozedur (= Unterprogramm). Entsprechende Parameter lassen sich in Klammern als Variablen übergeben.

END PROC

...bestimmt das Ende einer Prozedur. Damit wird sie zum Unterprogramm und läßt sich jederzeit mit EXEC aufrufen. Achtung: Vermeiden Sie den Aufruf mit CALL, sonst erhalten Sie die Fehlermeldung »END PROC WITHOUT EXEC«.

CALL name[(var1, var2)]

Parameter: name = Prozedurenname, var1/var2 = beliebige Variablen (Zahlen, Strings usw.).

...ruft das mit PROC definierte Unterprogramm auf, das aber nicht mit END PROC abschließen darf! Auch hier lassen sich Werte durch Variablen übergeben.

```
10 CALL FARBE (A*7,15,Z,"FARBE")
20 END
30 PROC FARBE (D,E,F,H$)
40 COLOR D,E,F: CENTER H$
50 GOTO 20
```

EXEC name[(var1,var2)]

Parameter: name = Prozedur, var1/var2 = Variablenliste.

...aktiviert die Prozedur als Unterprogramm. Bei END PROC springt der Befehlsinterpreter zum Hauptprogramm zurück.

```
10 INPUT A: IF A=1 THEN CALL AENDERN
15 END
20 PROC AENDERN
30 INPUT A,B,C: EXEC FARBE(A,B+1,C)
40 END
50 PROC FARBE(X,Y,Z)
60 COLOR X,Y,Z: END PROC
```

SGOTO zn

Parameter: zn = berechenbare Zeilennummer (0 bis 63999)

...funktioniert wie GOTO im Basic 2.0 – allerdings kann die Zeilennummer auch als Rechenformel mit Variablen angegeben werden.

```
10 INPUT A: SGOTO 100+A*10
```

POP x

Parameter: x = Unterprogrammart:

- 1: GOSUB/RETURN,
- 2: EXEC/END PROC,
- 3: REPEAT/UNTIL,
- 4: LOOP/ENDLOOP

Korrektes Verlassen eines Unterprogramms oder einer Schleife (Stapelzeiger wird korrigiert).

```
10 GOSUB 100
100 GETKEY A$
110 IF A$="X" THEN POP 1: GOTO 200
120 X$ = A$: RETURN
200 PRINT "ENDE": END
```

RESET zn

Parameter: zn = Zeilennummer.

...setzt den DATA-Zeiger auf die Zeilennummer zn. Damit lassen sich gezielt Daten innerhalb des Programms lesen.

CASE OF wert – WHEN [wert] – OTHERWISE – ENDCASE

Parameter: wert = Zahlen oder Strings (müssen bei CASE OF und WHEN übereinstimmen).

Der hinter CASE OF eingetragene Wert wird so lange mit dem hinter WHEN verglichen, bis das Programm eine Übereinstimmung feststellt. Dann kommen alle Befehle hinter WHEN bis zum nächsten WHEN, OTHERWISE oder ENDCASE zur Ausführung. Sind alle Befehle hinter WHEN erledigt, springt der Interpreter zu ENDCASE und beendet die Abfrage. Stimmt der Wert hinter WHEN nicht mit dem von CASE OF überein, verzweigt das Programm zu OTHERWISE und aktiviert alle nachfolgenden Befehle bis END CASE. Sollen mehrere Werte hinter WHEN überprüft werden, muß man sie durch Kommas trennen.

```
10 REPEAT
20 GETKEY A$
30 CASE OF A$
40 WHEN "1","2","3": CLS
50 SGOTO VAL(A$) * 10 + 100
60 WHEN "X": PRINT "ENDE"
70 OTHERWISE: PRINT "UNGUELTIG"
80 ENDCASE
90 UNTIL A$="X": END
```

III. Fehlerbehandlung

HELP

Parameter: keine

Gibt's im Programmodus einen Fehler, erscheint die falsche Zeile nach Eingabe von HELP auf dem Bildschirm. Die Anweisung, die den Fehler verursacht hat, wird revers gezeigt.

ON ERROR GOTO zn

Parameter: zn = Zeilennummer

Sprung zu einem Unterprogramm, das bei einem Fehler aktiviert werden soll.

```
10 ON ERROR GOTO 50
20 FOR X= -10 TO 10
30 PRINT 1/X: NEXT
40 END
50 PRINT "FEHLER"
```

In den Systemvariablen ERRLN findet man die Nummer der Fehlerzeile, ERRN speichert die Fehlernummer. Fehlermeldungen im Klartext fragt man mit PRINT ERR$(ERRN) ab:

```
60 PRINT "IN ZEILE"; ERRLN
70 PRINT "TYP: "; ERR$(ERRN)
```

NO ERROR

Parameter: keine

...schaltet den ON-ERROR-Modus ab. Jetzt gelten wieder die normalen Fehlermeldungen des Basic 2.0.

RESUME [NEXT]

Parameter: keine

...macht nach einem Fehler im Programm weiter, wenn der ON-ERROR-Modus aktiviert war. Im Fehlerbehandlungs-Programmteil müssen diese Befehle integriert sein: RESUME oder RESUME NEXT.

RESUME führt die fehlerhafte Anweisung nochmals aus; RESUME NEXT aktiviert den Befehl hinter der Anweisung, die den Fehler verursachte.

IV. Eingabe- und Ausgabebefehle

FETCH a$,l,x$

Parameter: a$ = definierte Zeichenfolge, z.B. " 0123456789", l = Anzahl der einzugebenden Zeichen (von 0 bis 88), x$ = Variablenname für die Eingabe.

Verbesserte INPUT-Routine des Basic 2.0: Bei der Eingabe werden nur solche Zeichen akzeptiert, die A$ enthalten. Damit lassen sich z.B. Steuerzeichen (Cursor-Tasten, <CLR/HOME> usw.) verbieten – lediglich <DEL> bleibt aktiv, um Eingaben zu löschen. Außerdem können maximal l Zeichen eingetragen werden. Mit <RETURN> schließt man die Eingabe ab; sie wird in X$ gespeichert.

```
10 FETCH "123456789",2,X$
20 PRINT VAL(X$)
```

GETKEY a$

Parameter: a$ = Stringvariable

Programm wartet auf einen Tastendruck und weist dessen Wert der Variablen A$ zu.

```
10 PRINT "TASTE DRUECKEN!"
20 GETKEY T$
30 PRINT T$
```

LOCATE x,y [,Ausgabeliste]

Parameter: x = Textspalte (0 bis 39), y = Textzeile (0 bis 24), Ausgabeliste = Stringvariable oder Klartext in Anführungszeichen.

...setzt den Cursor auf die angegebene Spalte bzw. Zeile des Textbildschirms und plaziert dort den gewünschten Text.

```
10 LOCATE 20,12,"BILDSCHIRMMITTE"
```

oder als Digitaluhr-Simulation:

```
10 CLS
20 TI$="120000"
30 LOCATE 32,0
40 PRINTCHR$(18)LEFT$(TI$,2)+":"+
MID$(TI$,3,2)+":"+RIGHT$(TI$,2)
50 GOTO 30
```

CENTER a$

Parameter: a$ = Stringvariable

...plaziert den Variablentext in der Bildschirmmitte. Ist er länger als 41 Zeichen, entfällt die Zentrierung.

```
5 CLS
10 LOCATE 18,12
20 CENTER TI$
30 GOTO 10
```

CHRLO

Parameter: keine

...stellt den Kleinschriftzeichensatz ein (wie PRINT CHR$(14).

CHRHI

...aktiviert wieder die Großschrift und Blockgrafik-Zeichenmuster (entspricht PRINT CHR$(142)) Die Tastenkombination <CBM SHIFT> ist außer Funktion gesetzt.

CLS

...löscht den Textbildschirm (wie PRINT CHR$(147)).

SPELL a$,n

Parameter: a$ = Text, den man ausgeben will, n = Geschwindigkeit (0 = am schnellsten bis 255 = am langsamsten).

...verzögert die Bildschirmausgabe von Text A$ um den Faktor n.

```
10 CLS
20 LOCATE 10,12
30 SPELL "BEISPIELTEXT",200
40 GETKEY T$
```

PRESS r,a$

Parameter: r = Richtung (1 bis 8), a$ = auszugebender Text.

...schreibt den Inhalt von A$ in der Richtung r auf den Bildschirm:

- r = 1: gradlinig von unten nach oben,
- r = 2: diagonal nach rechts oben,
- r = 3: gradlinig von links nach rechts,
- r = 4: diagonal nach rechts unten,
- r = 5: von oben nach unten,
- r = 6: diagonal nach links unten,
- r = 7: gradlinig von rechts nach links,
- r = 8: diagonal nach links oben.

```
10 CLS
20 A$="BEISPIEL"
30 FOR I=1 TO 8
40 LOCATE 18,12
50 PRESS I,A$
60 NEXT
```

V. Bildschirmgrafik

COLOR sf, rf, hf

Parameter: sf = Farbe der Schriftzeichen (0 bis 15), rf = Bildschirmrahmen (0 bis 15), hf = Hintergrundfarbe (0 bis 15).

...legt die Farbwerte der Bildschirmausgabe fest.

FCOL s, z, b, h, f

Parameter: s = Spalte (0 bis 39), z = Zeile (0 bis 24), b = Breite (1 bis 40), h = Höhe (1 bis 25), f = Farbe (0 bis 15).

...füllt definierte Bildschirmbereiche mit der Vordergrundfarbe F. S und Z sind die Koordinaten oben links, B gibt die Breite vom Punkt S/Z nach rechts an. H bestimmt die Höhe ab S/Z nach unten.

```
FCOL 0,0,40,25,1
```

Achtung: Bei gelöschtem Bildschirm tut sich offiziell nichts, da dieser Befehl nur Schrift- und Grafikzeichen im Bildvordergrund umfärbt!

FCHR s, z, b, h, c

Parameter: z, s, b, h = s. FCOL, c = Bildschirmcode des gewünschten Zeichens.

...überflutet vorbestimmte Bildschirmsegmente mit dem Zeichen c.

```
FCHAR 3,5,7,9,1
```

FILL s, z, b, h, c, f

Parameter: s, z, b, h = s. FCOL, c = s. FCHAR, f = Farbe (0 bis 15).

...ist die Zusammenfassung der Befehle FCOL und FCHAR: Ein definierter Bildausschnitt wird mit farbigen Zeichen gefüllt.

SCRINV s, z, b, h

Parameter: s, z, b, h = s. FCOL.

Invertieren eines Bildschirmbereichs (positive Zeichen werden negativ und umgekehrt).

```
SCRINV 1,1,38,11
```

Jetzt sieht man z.B ein Grafik-Window auch auf gelöschtem Bildschirm!

SCOPY s, z, b, h, ns, nz

Parameter: s, z, b, h = s. FCOL, ns = neue Spalte, nz = neue Zeile.

...kopiert einen Bildschirmbereich, der in der linken oberen Ecke S/Z beginnt, an die neue Koordinate NS/NZ.

```
SCOPY 4,5,2,3,10,11
```

CHANGE

...erzeugt einen Hilfsbildschirm im RAM-Bereich von $F000 bis $F3FF. Das Farb-RAM liegt ab $F400 bis $F7FF im Speicher. Der Bildinhalt des aktuellen Screens wird darin gesichert. Erneutes CHANGE stellt den Urzustand wieder her.

Achtung: Wenn Sie mit hochauflösender Grafik arbeiten, sollten Sie diesen Befehl nicht anwenden, da sich sonst die Speicherbereiche überschneiden.

SUP s, z, b, h, m
Parameter: s, z, b, h = s. FCOL, m = Modus (0 = ohne Bildumlauf, 1 = mit).
...scrollt den Bildausschnitt S, Z, B, H um eins nach oben. Ist M = 1, erscheinen Zeichen, die in der obersten Zeile verschwunden sind, wieder unten. Bei M = 0 bleiben weggescrollte Zeilen verschwunden. Die unterste Zeile füllt sich dann mit Leerzeichen.

```
10 REPEAT
20 SUP 0,0,20,12,1
30 SUP 20,13,20,12,1
40 GET A$
50 UNTIL A$<>""
60 FOR I=1 TO 25
70 SUP 0,0,40,25,0
80 GETKEY A$
90 NEXT
```

SDOWN s, z, b, h, m
Parameter: s. SUP
...funktioniert wie SUP, allerdings scrollt der Bildschirm jetzt nach unten.

```
10 LOOP
20 SDOWN 0,0,40,13,1
30 SUP 0,13,40,12,1
40 END LOOP
```

SRIGHT s, z, b, h, m
Parameter: s. SUP, SDOWN
...scrollt den Bildschirm nach rechts.

```
10 CLS
20 LOCATE 13,13,"SCROLL-TEST"
30 LOOP
40 SRIGHT 10,13,20,1,1
50 FOR T=1 TO 100: NEXT
60 END LOOP
```

SLEFT s, z, b, h, m
Parameter: s. SUP.
...verschiebt den Text auf dem Bildschirm kontinuierlich nach links.

```
SLEFT 0,0,40,25,0
```

FLASH [f, g]
Parameter: f = Farbe (0 bis 15), g = Geschwindigkeit im 1/60stel-Sekunden-Takt (1 = sehr schnell, 255 = ganz langsam.

Alle Zeichen mit der Farbe F werden auf dem Bildschirm wechselweise revers und wieder normal gezeigt. FLASH ohne Parameter stellt den Modus wieder ab.

```
10 COLOR 0,15,15
20 FILL 0,0,40,25,32,15
30 LOCATE 18,12,"FLASH"
40 FLASH 0,70
50 GETKEY A$
60 FLASH
```

VI. Floppybefehle

DLOAD
...lädt ein Basic-Programm von Diskette.

```
DLOAD "TESTNAME"
```

Die gewohnte Endung »,8« der Basic-2.0-Ladeanweisung entfällt!

DSAVE
...speichert ein Basic-Programm auf Disk.

DVERIFY
...vergleicht das Basic-Programm im Speicher mit dem gleichnamigen auf Diskette.

DMERGE
...integriert die Basic-Zeilen eines von Diskette zu ladenden Programms in das im Speicher. Achtung: Besitzt das nachgeladene Listing identische Zeilennummern, werden die alten Basic-Zeilen durch die neuen ersetzt!

CHAIN
...lädt und startet ein Basic-Programm von Disk.

```
CHAIN "TESTNAME"
```

GLOAD
...holt ein Hires-Grafikbild inkl. Farb-RAM in den Computer.

```
GLOAD "GRAFIK"
```

GSAVE
...speichert den Grafikschirm inkl. Farb-RAM als Datei auf Disk.

BLOAD »NAME« [,sa]
Parameter: sa = Startadresse.
...lädt einen Datenbereich absolut (wie mit »,8,1« in Basic 2.0), z.B. Assemblerprogramme. Wird SA angegeben,

```
BLOAD "UTILITY"
```

Normal wird die am Dateibeginn eingetragene Ladeadresse berücksichtigt. Durch den Parameter SA kann man jedoch beliebige Speicherstellen angeben, z.B.:

```
BLOAD "ZEICHENSATZ",12288
```

BSAVE »NAME«, sa, ea + 1
Parameter: sa = Startadresse, ea = Endadresse der Datei.
...sichert den Bereich im Computerspeicher zwischen SA und EA+1 (z.B. Maschinensprache-Code). Solche Dateien lassen sich mit BLOAD laden.

```
BSAVE "ASSEMBLERCODE", $C000, $C4FF
```

Achtung: Mit Special Basic erzeugte Hires-Grafiken lassen sich nicht mit dieser Anweisung sichern! BSAVE speichert immer das ROM ab $E000, aber nicht das Grafik-RAM. Das macht nur GSAVE. Wer Special-Basic-Grafiken im Hi-Eddi-Format speichern möchte, benutzt unser Utility »Special-Saver« (s. Rubrik »Tips & Tricks« in diesem Heft!).

CLOAD
...holt einen Zeichensatz in den Computer, der mit CSAVE gespeichert wurde.

```
CLOAD "CHAR1"
```

CSAVE
...sichert neue Zeichensatzmuster auf Disk.

DIR
...bringt das Inhaltsverzeichnis der aktuellen Diskette im Laufwerk auf den Bildschirm. <SHIFT> hält die Ausgabe an, <RUN/STOP> bricht sie ab.

DISK A$
Parameter: a$ = Floppykommando als Zeichenkette.
...gibt die Floppyanweisung an die Diskettenstation weiter.

```
DISK "N:DATENDISK,D1"
```

entspricht dem umständlicheren Befehl des Basic 2.0 mit OPEN- und CLOSE-Anweisungen.

ERROR
...liest den Fehlerkanal (wenn Floppy-LED blinkt!) und zeigt den Status im Klartext, z.B.:

```
62, FILE NOT FOUND, 00, 00
```

VII. Druckerbefehle

TYPE A$
Parameter: a$ = Text, der gedruckt werden soll.

Diese Anweisung funktioniert wie PRINT, die Daten werden aber nicht zum Bildschirm, sondern zum seriell angeschlossenen Drucker geschickt.

```
TYPE "HALLO, TEST!"
TYPE 1234567
```

TEXTCOPY
...druckt den aktuellen Textbildschirm (Low Resolution, Lores) zentriert in der Papiermitte.

GRAPHCOPY
...schickt die Daten einer hochauflösenden Grafik (High Resolution, Hires) zum Drucker und gibt sie zentriert aus.

Selbstverständlich lassen sich auch Bilder aller bekannten Hires-Zeichenprogramme des C64 laden und verarbeiten (z.B. Hi-Eddi, Starpainter, OCP Art Studio usw.). Zwei Dinge muß man allerdings beachten: Die Grafikdateien dürfen nicht mehr als maximal 33 Blocks auf Diskette belegen (Multicolorgrafiken à la Koala-Painter, Paint Magic oder Amica Paint sind tabu!) und müssen in den Hires-Grafikbereich von Special Basic geladen werden: $E000 (57344):

```
BLOAD "(Fremdgrafik)",$E000
```

Alle Druckerbefehle berücksichtigen die Geräteadresse 4 und seriell mit dem Computer verbundene Drucker (z.B. per Hardware-Interface an der Centronics-Schnittstelle).

VIII. Zeichensatz

Wenn Sie mit geänderten Zeichensätzen arbeiten, müssen Sie auf Hires-Grafik verzichten: Beide Modi benutzen denselben Speicherbereich ab $E000.

NCHAR
...kopiert die Originalmuster des Commodore-Zeichensatzes vom ROM ($D000) ins RAM ($E000), wo sie sich beliebig ändern lassen.

SWITCH m

Parameter: m = Modus (0 = Originalzeichensatz, 1 = neue Zeichenmuster).

Bei M = 1 wird der Speicher des C 64 neu eingerichet: Der aktuelle Zeichensatz beginnt jetzt bei \$E000, der Bildschirmspeicher wird nach \$CC00 (52224) verschoben. Ab Adresse \$C000 (49152) können Sprite-Blöcke liegen. Ist M = 0, ruft man die Standardeinstellung auf: Zeichensatz ab ROM-Bereich \$D000 (53248), Bildschirm bei \$0400 (1024).

```
10 NCHAR: SWITCH 1
20 FOR I=57344 TO 57351
30 READ D: POKE I,D
40 NEXT
50 CLS: PRINT CHR$(64)
60 DATA 28,34,77,81
70 DATA 81,77,34,28
```

Das Listing verwandelt den Klammeraffen ins Copyright-Zeichen und zeigt es in der linken oberen Bildschirmecke.

CHRINV c, gk

Parameter: c = Bildschirmcode (0 bis 255), gk = Zeichensatzeinstellung (0 = Großschrift/Blockgrafik, 1 = Klein- und Großschrift).

...invertiert das Zeichen mit dem betreffenden Bildschirmcode im mit GK bestimmten Zeichensatzmodus.

```
10 NCHAR: SWITCH 1
20 CHRINV 1,0
```

...bringt den Buchstaben A (Bildschirmcode 1) revers.

TWIST c, gk, x

Parameter: c, gk = s. CHRINV, x = Anzahl, wie oft das Zeichen gedreht werden soll.

...rotiert das mit C und GK gewählte Zeichen x-mal um 90 Grad nach links.

```
10 NCHAR: SWITCH 1
20 FOR I=0 TO 255
30 TWIST I,0,1
40 NEXT
```

MIR c, gk

Parameter: c, gk = s. CHRINV.

Das Zeichen wird an der x-Achse gespiegelt und steht dann auf dem Kopf.

```
10 NCHAR: SWITCH 1
20 FOR I=0 TO 255
30 MIRX I,0
40 NEXT
```

MIRY c, gk

Parameter: c, gk = s. CHRINV

...spiegelt das Zeichen an der y-Achse. Es erscheint seitenverkehrt.

CREATE 0, c, gk

Parameter: c, gk = s. CHRINV.

Zeichen ändern. Diese Anweisung muß vor jeder Neudefinition mit CHAR stehen!

CHAR a\$ [,z]

Parameter: a\$ = Zeichenkette, die aus <*> (= Bit gesetzt) und <.> (= gelöschtes Bit), z = Byte-Zeile im Zeichenmuster (1 bis 8).

...ändert eine Bit-Zeile (= Byte) im Muster des mit CREATE gewählten Zeichens. Gibt man dahinter den Parameter Z an, betrifft die Änderung nur dieses Byte der Zeichenmatrix.

```
10 NCHAR: SWITCH 1
20 CREATE 0,5,0
30 CHAR "...***.."
40 CHAR ".*.***.*"
50 CHAR ".*..*..*"
60 CHAR ".*******"
70 CHAR "...***.."
80 CHAR "...***.."
90 CHAR "...*.*.."
100 CHAR ".***.***"
```

Statt E (Bildschirmcode 5) erscheint jetzt eine Figur.

```
10 NCHAR: SWITCH 1
20 CREATE 0,1,0
30 CHAR "********",1
40 CHAR "********"
50 CHAR "********",7
60 CHAR "********"
```

...stattet das A jeweils oben und unten mit einem Balken aus.

CHRCOPY c, gk, an, cn, gkn

Parameter: c, gk = s. CHRINV, an = Zeichensatz (0 = Original), (1 = neuer Zeichenmusterbereich), cn, gkn = neues Zeichen, das mit dem alten (c, gk) überschrieben wird.

...überträgt das Zeichenmuster eines Charakters aus dem neuen Zeichensatz in die Matrix des äquivalenten Originalzeichens.

```
NCHAR: SWITCH 1: CHRCOPY 1,1,1,0,0
```

...kopiert das kleine a in den Klammeraffen.

CHROR c1, gk1, an1, c2, gk2, an2, c, gk

Parameter: c1, gk1, an1, c2, gk2, an2 = s. CHRCOPY, c, gk = s. CHRINV.

Die per C1, GK1, AN1 und C2, GK2, AN2 definierten Zeichen werden ODER-verknüpft (alle gesetzten Bits sind sichtbar) und im Zeichen C, GK gespeichert.

```
10 NCHAR: SWITCH 1
20 CHROR 91,0,0,86,0,0,42,0
```

ändert <*> (Bildschirmcode 42).

CHRAND c1, kg, an1, c2, gk2, an2, c, gk

Parameter: c1, gk1, an1, c2, gk2, an2 = s. CHRCOPY, c, gk = s. CHRINV.

...aktiviert die UND-Verknüpfung der einzelnen Zeichenmuster (s. CHROR).

```
10 NCHAR: SWITCH 1
20 CHRAND 77,0,0,78,0,0,46,0
```

...wandelt z.B. den Punkt um (Bildschirmcode 46).

SCROLL r c, gk , a, m

Parameter: r = Bewegungsrichtung (U = aufwärts, R = rechts, D = runter, L = links).

Scrollt das durch C,GK bestimmte Zeichen A-mal in die durch den Kennbuchstaben R angegebene Richtung. Bei Modus M = 1 erscheinen bei der Screen-Begrenzung verschwundene Zeichen wieder auf der gegenüberliegenden Seite. Ist M = 0, wird die Spalte bzw. Zeile gelöscht.

```
5 CLS
10 NCHAR: SWITCH 1
20 FCHR 0,0,40,25,102
30 REPEAT
40 SCROLL D 102,0,1,1
50 GET A$
60 UNTIL A$="X"
```

EMPTY c, gk

Parameter: c, gk = s. CHRINV.

...löscht das durch C, GK bestimmte Zeichen.

```
NCHAR: SWITCH 1: EMPTY 1,0
```

CHAR\$(c, gk, an, z)

Parameter: c, gk, an = s. CHRCOPY, z = Byte-Zeile (1 bis 8).

...ist eine String-Funktion wie z.B. MID\$. Sie übergibt als Wert eine achtstellige Zeichenkette mit <*> bzw. <.> und definiert die Byte-Zeile Z des durch C, GK, AN bestimmten Zeichens.

Mit dieser Anweisung erhält man das gesamte Zeichenmuster (z.B. A):

```
FOR I=1 TO 8: PRINT CHAR$(1,0,0,I): NEXT
```

Beispiel ":

```
CREATE 0,10,1
CHAR CHAR$(10,1,1,2),1
```

IX. Sprite-Befehle

DEFMOB n, b, f, p, m

Parameter: n = Sprite-Nummer (1 bis 8), b = Sprite-Block (0 bis 255), f = Farbe (0 bis 15), p = Sprite-Priorität (0 = vor, 1 = unter dem Hintergrund), m = Modus (0 = Hires-Sprite [24 x 21 Pixel, einfarbig], 1 = Multicolor-Sprite [12 x 21 Bildpunkte, dreifarbig]).

...Eigenschaften des Sprites bestimmen und einschalten.

Achtung: Wird Hires-Grafik oder ein geänderter Zeichensatz als Hintergrund verwendet, darf man die DEFMOB-Anweisung erst nach Einschalten der Grafik anwenden!

MULTI f1, f2

Parameter: f1 = Multicolorfarbe 2, f2 = Mulitcolorfarbe 3. Farbe 1 wird bereits bei DEFMOB festgelegt.

...wählt die Zusatzfarben für Multicolor-Sprites.

MOBEX n, x, y

Parameter: n = Sprite-Nummer (1 bis 8), x = horizontal vergrößern (0 = normal, 1 = doppelt), y = vertikale Ausdehnung (0 = normal, 1 = zweifach.

...dehnt das Sprite-Muster Nr. N in die gewünschte Richtung aus. Setzt man x und y auf 1, erhält man ein vierfach vergrößertes Sprite!

```
MOBEX 7,1,1
```

CLEAR [n]

Parameter: n = Sprite-Nummer (1 bis 8).

Sprite Nr. N wird abgeschaltet. CLEAR ohne Parameter schaltet alle Sprites ab.

MOBSET n, x, y

Parameter: n = Sprite-Nummer (1 bis 8), x = x-Bildschirmkoordinate (0 bis 511), y = y-Koordinate (0 bis 255).

...positioniert das Sprite N an gewünschter Stelle. Achtung: Die x- und y-Koordinaten 0,0 des Sprites entsprechen im Grafikmodus den Bildschirmpositionen 24/50.

```
MOBSET 1,200,180
```

MMOB n, x1, y1, x2, y2, g

Parameter: n = Sprite-Nummer (1 bis 8), x1/y1/x2/y2 = s. x, y bei MOBSET, g = Geschwindigkeit (0 = schnell bis 255 = langsam).

...bewegt ein Sprite mit dem Speed G von X1/Y1 nach X2/Y2.

```
MMOB 1,100,120,150,90,30
```

MOVE n, x, y, g

Parameter: n = Sprite-Nummer (1 bis 8), x/y, g = s. MOBSET.

...verschiebt ein Sprite von der aktuellen Position mit der Geschwindigkeit G zur Koordinate X/Y.

```
MOVE 2,300,150,100
```

CLRVIC

...setzt alle veränderten Sprite-Parameter (z.B. Position, Priorität, Vergrößerung usw.) wieder auf 0.

```
10 CLRVIC
20 ERASE 13,0: BLINV 13,0
30 DEFMOB 3,13,1,0,0
40 MOBEX 3,13,1,0,0
50 MOBSET 3,100,100
60 GETKEY A$
70 REPEAT
80 MOVE 3,400 * RND(0),250, * RND(0),20
90 GET A$
100 UNTIL A$<>""
110 MMOB 3,0,150,200,150,100
```

BUMP (n1 [, n2])

Parameter: n1, n2 = Sprite-Nummer (1 bis 8).

...wirkt wie die Funktion PEEK. Sie stellt fest, ob (bei BUMP (N)) das Sprite Nr. N mit dem Bildschirmvordergrund kollidiert oder zwei Sprites zusammengestoßen sind (BUMP n1, n2). Bei einem der genannten Crashes ist das Ergebnis immer 1, sonst 0.

```
PRINT BUMP (6,7)
IF BUMP(3) = 1 THEN 90
```

Wenn's nur um den Hinweis geht, ob ein bestimmtes Sprite mit einem beliebig anderen zusammenstößt, muß man n1 = n2 setzen, z.B. BUMP (3,3).

CREATE 1, b, d

Parameter: b = Sprite-Block (0 bis 255), d = Bereich (0 bis 2).

...definiert den gewünschten Sprite-Block B im Bereich D. Ist D = 0, sind die Speicheradressen von 0 bis 16383 ($3FFF) gemeint: Block 0 beginnt bei Speicherzelle 0, Sprite-Block 1 bei 64 usw. Diese Einstellung ist beim normalen Textbildschirm zwingend vorgeschrieben. Wurde Grafik oder ein neuer Zeichensatz aktiviert, muß D = 2 sein. Blöcke mit der Bereichsnummer 1 lassen sich nur durch Verschieben nutzen.

MOB a$ [, z]

Parameter: a$ = Zeichenkette, die eine Byte-Zeile des Sprites repräsentiert. Wie bei CHAR gilt <*> als gesetztes, <.> als gelöschtes Pixel. Bei Multicolor-Sprites werden die beiden Zusatzfarben 2 und 3 durch die Tastaturzeichen </> und <=> gekennzeichnet, <*> gilt nach wie vor als Farbe 1.

Das Funktionsprinzip entspricht dem Verfahren, das man zur Definition neuer Zeichen verwendet (CHAR). Der Unterschied: Die Editorfläche besteht nun aus 21 Zeilen 24 Bildpunkten.

```
10 CREATE 1,13,0
20 MOB DUP ("*",24)
30 FOR I=1 TO 19
40 MOB "*" + DUP (".",22) + "*"
50 NEXT
60 MOB DUP ("*",24)
70 DEFMOB 1,13,0,0,0
80 MOBEX 1,0,0
90 MOBSET 1,100,100
```

TURNMOB b, d, a

Parameter: b, d = s. CREATE, a = Anzahl, wie oft das Zeichen gedreht werden soll.

...rotiert den Sprite-Block B/D A-mal um 90 Grad.

```
TURNMOB 13,0,1
```

BLCOPY b, d, bn, dn

Parameter: b, d, bn, dn = s. CREATE.

...überträgt das Pixelmuster von Sprite-Block B/D ins Sprite BN/DN.

```
BLCOPY 13,0,14,0
```

BLOR b1, d1, b2, d2, b, d

Parameter: b1, d1, d2, b, d = s. CREATE.

ODER-Verknüpfung der Sprite-Muster B1/D2 und B2/D2. Das Ergebnis erscheint in Sprite-Block B/D.

```
BLOR 13,0,14,0,15,0
```

BLAND b1, d1, b2, d2, b, d

Parameter: s. CREATE

Die Sprite-Blöcke B1/D1 und B2/D2 werden UND-verknüpft. Das neue Muster kommt in den Block B/D.

ERASE b, d

Parameter: b, d = s. CREATE.

...löscht das alte Sprite-Muster des Blocks B/D.

```
ERASE 13,0
```

BLINV b, d

Parameter: s. ERASE.

...zeigt das Sprite-Muster des Blocks B/D revers.

```
BLINV 13,0
```

REFLECTX b, d

Parameter: s. ERASE.

...spiegelt den Sprite-Block an der x-Achse: Jetzt steht das Sprite auf dem Kopf.

```
REFLECTX 13,0
```

REFLECTY b, d, m

Parameter: b, d = s. ERASE, m = Modus (0 = Hires-, 1 = Multicolor-Sprite.

...dreht das Sprite an der y-Achse: Es erscheint seitenverkehrt.

```
REFLECTY 13,0,0
```

SCRMOB r, b, d, a, m

Parameter: r = Kennbuchstaben U, D, R oder L (s. SCROLL), b/d = s. CREATE, a = Anzahl, wie oft gescrollt werden soll, m = Modus (0 = ohne, 1 = mit Umlauf).

...funktioniert wie das Scrollen von Textzeichen (s. Beschreibung zu SCROLL).

```
SCRMOB R 13,0,2,1
```

SPRITE$ (b, d, z, m)

Parameter: b, d = s. CREATE, z = Zeile (1 bis 21), m = Modus (0 = Hires-, 1 = Multicolor-Sprite).

...besitzt die gleiche Funktion wie CHAR$. Damit wird eine 24stellige Zeichenkette von Zeile Z des Sprite-Blocks B/D gelesen und per PRINT ausgegeben.

```
FOR I=1 TO 21: PRINT SPRITE$(13,0,I,0): NEXT
```

oder

```
10 CREATE 1,13,0
20 MOB SPRITE$(14,0,1,0),21
```

X. Grafikbefehle

HIRES pf, hf

Parameter: pf = Pixelfarbe (0 bis 15), hf = Hintergrund (0 bis 15).

...löscht den hochauflösenden Grafikbildschirm, aktiviert den Grafikmodus und bestimmt die Farbe der Zeichenpixel im Vordergrund, sowie die des Bildhintergrunds. Der Grafikbildschirm wird im RAM-Bereich von $E000 (57344) bis $FFFF (65535) eingerichtet. Das Farb-RAM belegt jetzt den Speicher ab $CC00 (52224), die Sprites findet man ab $C000 (49152). Achtung: Hires-Grafik kollidiert mit einem geänderten Zeichensatz!

GRAPHIC [pf, hf]

Parameter: s. HIRES.

...schaltet den hochauflösenden Grafikmodus erneut ein, ohne den Bildschim zu löschen. Mit den Parameterangaben wird sie neu eingefärbt.

```
GRAPHIC: GETKEY A$
```

HICOL pf, hf

Parameter: pf, hf = s. HIRES.

Zeichen- und Hintergrundfarbe ändern.

```
HICOL 7,8
```

NRM

...zurück zum normalen Textbildschirm (Low Resolution, Lores). NRM wird automatisch beim Programmende oder nach Fehlermeldungen aktiviert.

PLOT x, y, zm

Parameter: x = horizontale Koordinate (0 bis 319), y = vertikale Koordinate (0 bis 199), zm = Zeichenmodus (0 = Pixel löschen, 1 = setzen, 2 = invertieren).

...setzt an den Koordinaten X/Y ein Pixel im Grafikbildschirm.

```
HIRES 0,1: PLOT 100,100: GETKEY A$
```

DRAW x1, y1, x2, y2, zm

Parameter: s. PLOT.

...zieht eine Linie von X1/Y1 nach X2/Y2.

```
HIRES 0,1: DRAW 40,50,100,170,1: GETKEY A$
```

REC x, y, b, h, zm

Parameter: x, y = s. PLOT, b = Breite, h = Höhe, zm = S. PLOT.

...zeichnet ein Rechteck bei den Koordinaten X/Y (links oben) mit Breite B und Höhe H in der aktuellen Pixelfarbe.

```
HIRES 0,1: REC 40,50,60,70,1: GETKEY A$
```

BOX x, y, b, h, zm

Parameter: s. REC.

...wie REC, aber das Rechteck färbt sich mit der Vordergrundfarbe.

```
HIRES 0,1: BOX 150,160,30,20,1: GETKEY A$
```

CIRCLE x, y, rx, ry, zm

Parameter: x/y = Koordinaten des Kreismittelpunkts, rx = Radius horizontal, ry = vertikal, zm = s. PLOT.

...zeichnet eine Ellipse mit dem Mittelpunkt X/Y und den Radien RX/RY. Sind beide Radiuswerte gleich, entsteht ein Kreis.

```
HIRES 0,1: CIRCLE 160,100,150,90,1: GETKEY A$
```

ARC x, y, rx, ry, sw, ew, a, zm

Parameter: x, y, rx, ry = s. CIRCLE, sw, ew = Start- und Endwinkel des Bogens, a = Abstand der Eckpunkte in Altgrad (0 bis 360), zm = s. PLOT.

...produziert einen Ellipsenabschnitt. Der Bogen wird vom Winkel SW bis EW gezeichnet. Winkel 0 ist dabei der Ursprung (oben Mitte).

```
10 HIRES 0,1
20 ARC 150,90,70,80,270,450,12,1
30 ARC 160,100,90,90,0,360,72,1
40 GETKEY $
```

Mit A wählt man den Abstand der Eckpunkte. Bei einer normalen Ellipse ist A = 12. Will man ein anderes Vieleck erzeugen, gilt folgende Formel:

```
A = 360 : Eckenanzahl
```

PAINT x, y, zm

Parameter: s. PLOT

...füllt eine umschlossene Grafikfläche mit der eingestellten Vordergrundfarbe. Ist der Modus ZM = 0, wird der Bereich bis zum nächsten gelöschten Pixel eliminiert. ZM = 2 (invertieren) ist nicht möglich.

```
10 HIRES 0,1
20 CIRCLE 160,100,100,20,1
30 PAINT 170,100,1
40 GETKEY A$
50 CIRCLE 160,100,150,90,1
60 PAINT 0,0,1
70 PAINT 160,100,0
80 GETKEY A$
```

SSHAPE a$, x, y, b, n

Parameter: a$ = Stringvariable (z.B. x$, gr$(1) usw.), x/y = s. PLOT, b/h = Breite und Höhe des Grafikausschnitts (SHAPE).

...speichert einen Teil des Grafikbildschirms ab X/Y mit Breite B und Höhe H in der Variablen A$. Ist B * H/8 + 4 größer als 255, gibt's eine Fehlermeldung: Der Teilgrafik-String darf nicht länger als 255 Zeichen sein!

```
SSHAPE S$,90,100,20,30
```

GSHAPE a$, x, y, zm

Parameter: a$ = s. SSHAPE, x/y, zm = Zeichenmodus.

...überträgt einen mit SSHAPE gespeicherten Grafikausschnitt an der Position X/Y in den aktuellen Grafikbildschirm.

Der Parameter ZM besitzt eine andere Bedeutung als bei PLOT (sowieso gelöschte Pixel des Shape werden nicht beachtet):
- zm = 0: löscht gesetzte Punkte,
- zm = 1: setzt installierte Pixel erneut,
- zm = 2: invertiert gesetzte Bildpunkte,
- zm = 3: vorher eingetragene Pixel werden gelöscht, ebenfalls vorher gelöschte Punkte!

```
10 HIRES 0,1
20 NCHAR: SSHAPE X$,0,0,24,24
30 GSHAPE X$,100,170,1
40 GETKEY A$
```

SBLOCK b, d, x ,y

Parameter: b, d = s. CREATE 1, x, y = s. PLOT.

...kopiert den Grafik-Shape ab X/Y in den Sprite-Block, der mit B/D definiert wurde.

```
SBLOCK 13,0,100,120
```

GBLOCK b, d, x, y, gx, gy, zm

Parameter: b, d = s. CREATE 1, x, y = s. PLOT, gx = horizontale Vergrößerung, gy = vertikal, zm = s. PLOT.

...fügt den Sprite-Block B/D an den Koordinaten X/Y in die aktuelle Grafik. Dabei läßt sich dieser Sprite-Shape in x- und y-Richtung dehnen: GX = 1 (normale Größe), GX = 2 (verdoppeln usw.). Das gilt selbstverständlich auch für den Parameter GY.

```
10 HIRES 0,1: NCHAR
20 SBLOCK 13,0,0,0
30 HIRES 0,1
40 GBLOCK 13,0,100,100,2,1,1
```

TEXT x, y, a$, gx, gy, a, zm

Parameter: x, y, zm = s. PLOT, a$ = String oder Zeichenkette im Klartext, gx/gy = Vergrößerung in x- und y-Richtung, a = Abstand der Zeichen in Pixel.

...positioniert beliebigen Text in der Grafik. X/Y definieren die linke obere Ecke des ersten Zeichens. Die Idealeinstellung für den Abstand: A = GX x 8. Erlaubte Steuerzeichen im String von TEXT: <RVS ON>, <RVS OFF> und <CTRL L> für Kleinschrift sowie <CTRL H> für den Großschrift-Blockgrafikmodus (Voreinstellung nach Programmstart).

```
10 HIRES 0,1
20 TEXT 10,20,"SPECIAL BASIC",2,2,16,1
30 GETKEY A$
```

WINDOW [az, ez]

Parameter: az = Anfangszeile (0 bis 199), ez = Endzeile (0 bis 200).

...splittet den Bildschirm in zwei Teile: Oben Hires-Grafik (von AZ bis EZ), darunter normaler Text-Screen. WINDOW ohne Parameter schaltet den Modus wieder ab.

```
WINDOW 8,103
```

REVERS

...invertiert den gesamten Grafikbildschirm.

SCNCLR

...löscht den Hires-Screen. Im Gegensatz zu HIRES werden aber weder aktuell eingestellte Farben verändert noch der Grafikmodus aktiviert.

GRCOL s, z, b, h, pf, hf

Parameter: s, z, b, h = s. FCOL, pf, hf = S. HIRES.

...färbt einen ausgewählten Grafikausschnitt um. Die Farbauflösung entspricht allerdings dem Textmodus: Jeweils ein 8 x 8-Pixelfeld erhält die gleiche Punkt- und Hintergrundfarbe.

```
10 HIRES 0,1: NCHAR
20 TEXT 10,160,"GRCOL-DEMO",2,2,16,1
30 LOOP
40 GRCOL 33*RND(0),18*RND(0),8,8,16*RND(0),
   16*RND(0)
50 END LOOP
```

TEST (x, y)

Parameter: x, y = s. PLOT.

...ist eine Funktion (wie z.B. SIN). Damit stellt man fest, ob der Koordinatenpunkt X/Y gesetzt (= 1) oder gelöscht (= 0) ist.

```
LOOP: PLOT 0,0,2: PRINT TEST(0,0): END LOOP
```

XI. Soundbefehle

VOL n

Parameter: n = Lautstärke (0, leise bis 15, laut).

...regelt das Klangvolumen des Soundchips (SID).

ENVELOPE st, a, d, s, r

Parameter: st = Stimme (1 bis 3), a/d/s/r = Klanghüllkurve (von 0 bis 15, A = Attack, D = Decay, S = Sustain, R = Release).

...bestimmt die ADSR-Kurve für die Stimme ST.

```
ENVELOPE 1,11,2,10,8
```

WAVE st, %xxxxxxxx

Parameter: st = Stimme (1 bis 3), %xxxxxxxx = 8 Bit für die Wellenform.

...stellt die Wellenform der Stimme ST ein. Achtung: Im Gegensatz zur sonst üblichen Bit-Darstellung liegt das niederwertigste Bit links außen, das höchste ganz rechts (also »01234567«). Die Bedeutung der Bits, wenn sie gesetzt (= 1) sind:
- 0 (1): Rauschen,
- 1 (2): Rechteck,
- 2 (4): Sägezahn,
- 3 (8): Dreieck,
- 4 (16): Test-Bit,
- 5 (32): Modulation,
- 6 (64): Synchronisation,
- 7 (128): Gate-Bit.

```
WAVE 1,%10000000
```

SOUND st, f [,l]

Parameter: st = Stimme (1 bis 3), f = Frequenz (0 bis 65535, l = Tonlänge (0 bis 65535, maximal 30 Sekunden).

SOUND spielt einen Ton in der Frequenz F und mit der Länge L für die Stimme ST. Läßt man L weg, wird nur F ins entsprechende SID-Register geschrieben.

```
SOUND 1,40000,10000
```

PULSE st, x

Parameter: st = s. SOUND, x = Zahl zwischen 0 und 4095.

Hat die Stimme Rechteck-Wellenform, läßt sich mit PULSE die Schwingungsfrequenz einstellen.

```
WAVE 1,%01000000: PULSE 1,250
```

CLRSID

...setzt den SID-Chip auf Standardwerte zurück. Jedes Musikprogramm sollte mit dieser Anweisung beginnen.

MUSIC g, a$

Parameter: g = Geschwindigkeit, a$ = Notenfolge als Einzelwert oder Zeichenkette.

...legt die Reihenfolge der Notenwerte fest, die dann per PLAY zum Klingen gebracht werden.
- Noten: C, D, E, F, G, A oder H,
- Oktave: 0 (tief) bis 7 (hoch),
- Notenlänge: 0 (kurz) bis 9 (lang).

Trägt man das Rautenkreuz < # > hinter einer Note ein, erhöht sie sich um einen Halbton.

```
MUSIC 100,"E#55C79G42"
```

Weitere Möglichkeiten, die Notenfolge zu beeinflussen:
- *st 1 bis 3: ändert die Stimme zu Beginn oder mitten im Notenstring. 1 ist voreingestellt.
- *pn 0 bis 9: Bei *p ohne Zahlenwert, klingt der letzte Ton aus und der nächste wird unmittelbar danach aktiviert. Mit Zahlenangabe wartet der Computer nochmals N Zeiteinheiten, bis der folgende erklingt.
- *n: Findet der Computer dieses Kommando, wird der gesamte davorliegende Musikteil ständig wiederholt. Vermeiden Sie diese Anweisung beim PLAY-Modus 1, da sonst das Musikstück niemals aufhört.

PLAY m

Parameter: m = Modus (0 bis 2).
- PLAY 0 stoppt die Musik,
- PLAY 1 startet sie und spielt unverdrossen bis zur nächsten Basic-Anweisung hinter der Notenkette durch,
- PLAY 2: ruft den Sound auf und läßt ihn im Interrupt spielen. Hier ist die Anweisung *N nützlich: Obwohl sich die Musik ständig wiederholt, ist der weitere Programmablauf nicht lahmgelegt.

BEEP x

Parameter: x = Anzahl (0 bis 255).

...erzeugt mit Stimme 3 einen Piepton, der x-mal wiederholt wird.

```
10 GETKEY A$: BEEP 1
20 IF A$="X" THEN BEEP 10: END
```

XII. Sonstige Befehle

SWAP v1, v2

Parameter: v1, v2 = Variablen, z.B. A, B$, C$ usw.

...tauscht die Inhalte von V1 und V2. Vorschrift: Die Variablennamen müssen vom gleichen Typ sein. Falsch wäre: SWAP A,B$ (numerisch und String!).

```
SWAP X$, B$(7)
```

DOKE a, w

Parameter: a = Adresse (0 bis 65535), w = 16-Bit-Wert (0 bis 65535).

...schreibt die Zahl W im Low-/Highbyte-Format in die Speicherzellen A und A+1.

```
DOKE 1024,300
```

BIT N a, n/BIT OFF a, n

Parameter: a = Adresse (0 bis 65535), n = Bit-Nummer (1 bis 8).

Ein- (= 1) oder Ausschalten (= 0) des Bit Nr. n in der Speicherstelle A. Hier stimmt die übliche Byte-Darstellung wieder: Bit 1 findet man rechts außen, Bit 8 steht ganz links.

```
10 CLS
20 POKE 1024,1
30 PAUSE 1
40 BIT ON 1024,8
50 PAUSE 1
60 BIT OFF 1024,8
70 GOTO 20
```

KILL

...entspricht der Reset-Anweisung SYS 64738 des Basic 2.0. Es erscheint die Einschaltmeldung von Special Basic. Programme werden gelöscht, lassen sich aber mit OLD erneut aktivieren.

BREAK ON/BREAK OFF

<RUN/STOP> ab- oder anschalten. Bei BREAK ON funktioniert auch die Tastenkombination <RUN/STOP RESTORE> nicht mehr!

FIX a

Parameter: a = Adresse (0 bis 65535.

...legt die Speicherstelle für den Befehl VARY fest.

```
FIX 2040
```

VARY [w1, w2, g]

Parameter: w1 = Wert 1 (0 bis 255), w2 = 2. Wert (0 bis 255), g = Geschwindigkeit (0 = schnell bis 255 = langsam).

Die per FIX bestimmte Adresse wechselt mit der Geschwindigkeit G/60stel Sekunden zwischen den Werten W1 und W2 (interruptgesteuerter Ablauf). VARY ohne Parameter schaltet den Modus wieder ab.

Anwendung für FIX und VARY: z.B. Bildschirmrahmen oder Hintergrund blinken lassen, Sprite-Blöcke wechseln usw.:

```
FIX 53280: VARY 1,2,100
```

PAUSE s

Parameter: s = Anzahl der Sekunden (0 bis 255).

Das Programm wartet s Sekunden, bevor es mit dem nächsten Befehl weitermacht.

COMSHIFT ON/OFF

...erlaubt oder verbietet die Umschaltung zwischen Groß- und Kleinschriftzeichensatz.

RAPID ON/OFF

...schaltet die Tastenwiederholungsfunktion an oder aus. Die Cursor-Tasten, <SPACE> und <INST/DEL> behalten ihre voreingestellte Wirkung.

SCREEN ON/OFF

Bildschirm ein- oder ausschalten. Bei deaktiviertem Screen füllt sich der gesamte Bereich mit der aktuellen Rahmenfarbe.

```
10 SCREEN OFF
20 PAUSE 5
30 SCREEN ON
```

LOCAL a, b%, c$...

Parameter: a, b%, c$ = beliebig viele Variablennamen.

...legt lokale Variablen (z.B. für eine Prozedur PROC) fest. Die Variablenwerte darf man jederzeit ändern, da der ursprüngliche Inhalt zwischengespeichert wird (s. GLOBAL).

Achtung: Definiert man mehr als 230 LOCAL-Variablen gleichzeitig, entsteht der Fehler OUT OF MEMORY.

```
10 I=9: A$="GLOBAL": Y%= -14
20 PRINT "GLOBALE WERTE";I;A$;Y%
30 LOCAL I: LOCAL A$;Y%
40 I=0: A$="LOCAL": Y%=7
50 PRINT "LOKALE WERTE";I;A$;Y%
```

GLOBAL a, b%, c$

Parameter: a, b%, c$ = s. LOCAL.

...reaktiviert die Variablen wieder mit den alten Werten, bevor sie mit LOCAL verändert wurden. Wenn Sie bei GLOBAL einen Variablennamen angeben, der zuvor nicht per LOCAL definiert wurde, kommt die Fehlermeldung UNKNOWN VARIABLE ERROR.

Hängen Sie beide folgenden Zeilen ans Demolisting zu LOCAL:

```
60 GLOBAL Y%,I,A$
```

```
70 PRINT "GLOBALE WERTE";I;A$;Y%
```

GO 64

...löscht den gesamten Speicher. Dabei verflüchtigen sich auch alle mit Special Basic generierten Basic- oder Assembler-Programme. Jetzt befindet sich der C64 wieder im Einschaltzustand (Basic 2.0).

LOMEM a

Parameter: a = Speicheruntergrenze (0 bis 65535).

Die Untergrenze für ein Basic-Programm wird festgelegt. Normalerweise ist dies 2049 ($0801): Die Adressen sollten in sinnvollen Bereichen liegen. Zusätzlich führt der Computer NEW aus (aktuelle Basic-Programme werden gelöscht!).

```
LOMEM 16384
```

HIMEM a

Parameter: a = oberste Speichergrenze (0 bis 65535).

...legt die höchste Adresse des Basic-Bereichs fest. Achtung: Bei Special Basic ist dies Speicherzelle 32767 ($7FFF)! Höhere Werte bringen die Basic-Erweiterung zum Absturz. Sinnvoll ist aber, HIMEM runterzusetzen, um z.B. Maschinenprogramme oder Datenbereiche zu schützen:

```
HIMEM 24576
```

Der Bereich von 24576 bis 32767 steht zur freien Verfügung und wird nicht von Basic-Variablen überschrieben. Das freie Basic-RAM hat dann allerdings nur noch 22 528 Byte.

CLREOL

... = CLear to End Of Line: löscht den Rest der Bildschirmzeile und die nächste ab Cursor-Position. Der Befehl funktioniert nur im Direktmodus. Holen Sie z.B. ein Programmlisting auf den Screen und bewegen Sie den Cursor auf den Beginn einer beliebigen Basic-Zeile. Überschreiben Sie die dort plazierten Bildschirmcodes mit

```
CLREOL:
```

(Doppelpunkt nicht vergessen) und drücken Sie <RETURN>. Der Befehl macht erst Sinn, wenn Sie ihn auf eine Funktionstaste legen (z.B. <F1>) und in einem Basic-Listing per Tastendruck fehlerhafte Zeilen löschen möchten.

BCKGND [hf1, hf2, hf3, hf4]

Parameter: hf1 bis hf4 = Hintergrundfarben (0 bis 15).

...schaltet vier verschiedene Hintergrundfarben. In diesem Modus stehen nur noch die Bildschirm-Codes 0 bis 63 zur Verfügung (also Ziffern, Buchstaben und gewisse Sonderzeichen). Ab sofort gilt der »Extended Colormode« (erweiterter Farbmodus) des C64: Die Bildschirmcodes 0 bis 63 erscheinen in der Hintergrundfarbe 1, die von 64 bis 127 mit Nr. 2. Farbe 3 gilt jetzt für die Codes von 128 bis 191, der Rest (192 bis 255) kommt in Farbe 4. BCKGND ohne Parameter deaktiviert den Modus.

```
10 BCKGND 15,1,2,3
20 POKE 1024,1
30 POKE 1025,65
40 POKE 1026,129
50 POKE 1027,193
```

Achtung: Extended Colormode und Hires-Grafik lassen sich nicht gleichzeitig aktivieren!

[[/]]

Zwischen den vier eckigen Klammern darf beliebiger Kommentar stehen, der mehrere Listingzeilen umfaßt.

```
10 [[
20 DAS IST EIN
30 TEST FUER DIE
40 ECKIGEN
50 KLAMMERN ]]
60 FOR I=1 TO 10
70 PRINT I: [[DRUCKT AUS]]: NEXT
80 [[ENDE]]
```

Funktionen

YPOS

...liefert die aktuelle Zeilenposition des Cursors im Textbildschirm.

```
FCHR 0,YPOS,40,25-YPOS,32
```

PENX

...übermittelt die horizontale Spalte (x-Position) eines angeschlossenen Lichtgriffels. Will man die x-Koordinate im Grafikbildschirm bestimmen, gilt diese Formel:

```
X=(PENX-40)*2
```

PENY

...s. PENX, allerdings dreht sich hier alles um die vertikale Position (y-Richtung).

INKEY

...stellt fest, welche Funktionstaste gedrückt wurde und speichert deren Nummer. Keine Taste gedrückt = 0.

ERRLN

...übergibt die Zeilennummer, in der das Programm den zuletzt registrierten Fehler entdeckt hat (s. ON ERROR).

```
PRINT ERRLN
```

ERRN

...registriert die Fehlernummer.

ERR$(n)

Parameter: n = Fehlercode (1 bis 42)

...bringt die Fehlermeldung Nr. n im Klartext.

```
10 FOR F=1 TO 42
20 PRINT F;ERR$(F)
30 NEXT
```

ASCII(c)

Parameter: c = Bildschirmcode 0 bis 255).

...konvertiert den Code des Bildschirmzeichens C in ASCII-Code, wie er bei CHR$() weitergegeben wird. Achtung: Bei allen Bildschirmcodes über 127 (reverse Zeichen) erhält man lediglich den CHR$-Wert des normalen Zeichens!

```
10 FOR I=0 TO 255
20 PRINT I,ASCII(I)
30 NEXT
```

BSC(a)

Parameter: a = ASCII-Code (0 bis 255)

Der umgekehrte Weg: CHR$-Werte werden nun als Bildschirm-Codezahlen ausgegeben.

```
10 FOR I=0 TO 255
20 PRINT I,BSC(I)
30 NEXT
```

DEC(A$)

Parameter: a$ = Umrechnungswert als Zeichenkette.

Damit lassen sich Hex- oder achtstellige Binärzahlen als Dezimalzahl anzeigen.

```
PRINT DEC("10110100")
PRINT DEC("FCE2")
```

HEX$(d)

Parameter: d = Dezimalzahl (0 bis 65535).

...gibt eine Dezimalzahl als Hexadezimalwert aus.

```
PRINT D, HEX$(57344)
E000
```

BIN$(d)

Parameter: d = Dezimalzahl (0 bis 255).

...rechnet eine Dezimalzahl als binäre Zeichenkette um.

```
PRINT BIN$(128)
10000000
```

ROW(a, b)

Parameter: a = Anfangs-ASCII-Code (0 bis 255), b = Endcode.

...stellt eine Zeichenkette zusammen, die ASCII-Code A bis ASCII-Code B enhält.

```
10 FETCH ROW(65,90)+" ",5,A$
20 PRINT "ES WURDE";A$;
30 PRINT "EINGEGEBEN"
```

DUP (a$, x)

Parameter: a$ = Zeichenkette, die vervielfacht werden soll, x = wie oft.

...dupliziert x-mal den String A$. Achtung: Das Produkt (A$ x X) darf eine Gesamtlänge von 255 Zeichen nicht übersteigen!

```
PRINT DUP("*-",20)
```

INSERT (a$,b,c$)

Parameter: a$ = einzufügender String, b = Position, an der eingepaßt werden soll, c$ = Zeichenkette, in die man einfügen will.

Die Gesamtlänge des zusammengesetzten Strings darf 255 Zeichen nicht überschreiten.

INST (a$,b,c$)

Parameter: a$, b = s. INSERT, c$ = ursprünglicher String.

...funktioniert wie INSERT, aber die alten Zeichen ab Position B werden überschrieben. Achtung: Die einzufügende Zeichenkette A$ darf nicht länger als C$ sein!

PLACE (a$,b$ [,c])

Parameter: a$ = zu suchender String, b$ = Zeichenkette, in der gesucht werden soll, c = Suchposition (< > 0).

In B$ wird nach A$ geforscht: entweder ab dem ersten Zeichen von B$ oder ab Position C.

USING (b$, z)

Parameter: b$ = Zeichenkette, die das Format für Zahlenausgaben definiert, z = entsprechende Zahl.

```
10 DZ$="# # # # . #,# "
20 LOOP
30 X = 1000*RND(0)
40 PRINT USING (DZ$,X)
50 END LOOP
```

Für jedes < # > reserviert der Computer eine Ziffer der Gesamtzahl Z. Entstehen mehr Nachkommastellen als in B$ vorgesehen, werden sie abgeschnitten. Sind's weniger, füllt sie das Programm mit Nullen auf. Statt < # > darf man jedes andere Zeichen benutzen, der Dezimalpunkt ist aber zwingend vorgeschrieben!

BY (s,z)

Parameter: s = Spalte (0 bis 39), z = Zeile (0 bis 24).

...setzt den Cursor im Textmodus an die per S und Z eingestellte Bildschirmposition.

DEEK (a)

Parameter: a = Speicheradresse (0 bis 65535).

...bringt eine 16-Bit-Zahl, die als Low- und Highbyte in a bzw. a+1 gespeichert ist. Den Beginn des Basic-Speichers findet man z.B. mit:

```
PRINT DEEK(43)
```

In Basic 2.0 ist's umständlicher:

```
PRINT PEEK(43) + 256 * PEEK(44)
```

CEEK (a)

Parameter: s. DEEK.

...entspricht der PEEK-Funktion des Basic 2.0, allerdings liest CEEK grundsätzlich das RAM unterm ROM.

CHECK (g)

Parameter: g = Gerätenummer.

...dient als Test, ob das Gerät G angeschlossen ist. Falls ja, erscheint 1, sonst 0.

```
10 LOOP: EXIT IF CHECK(8) = 1
20 PRINT "FLOPPY EINSCHALTEN!"
30 GETKEY A$: END LOOP
40 PRINT "OK."
```

ROOT (x,y)

Parameter: x = irgendeine Zahl, die ungleich 0 ist, y = dto, kann aber auch »0« sein.

...berechnet die xte Wurzel aus Y.

```
PRINT ROOT(3,27)
```

ROUND (a,b)

Parameter: a = Anzahl der Stellen, b = beliebige Zahl.

...rundet die Zahl B auf A Nachkommastellen. Ist A = 0, entfallen die Stellen nach dem Komma.

```
PRINT ROUND (123.456,2)
```

EVAL (a$)

Parameter: a$ = Zeichenkette, die Rechenaufgaben, aber noch keine Ergebnisse enthält (z.B. »56 * 65«, »SIN(9)«, »A + B« usw.).

EVAL gibt das Ergebnis aus.

```
5 N=1567.66
10 A$="N*1.15"
20 PRINT N"INKL. MWST=";
30 PRINT ROUND(EVAL (A$),2)
```

...berechnet die Mehrwertsteuer des Nettobetrags N und bringt das Bruttoergebnis, gerundet auf zwei Stellen hinter dem Komma.

EXOR (x,y)

Parameter: x und y = 16-Bit-Integer-Werte (zwischen -32768 bis 32767).

...verknüpft die Zahlen X und Y EXKLUSIV-ODER miteinander.

```
POKE 1024, EXOR (PEEK(1024),128)
```

...invertiert das Zeichen in der linken obersten Bildschirmecke.

JOY (n)

Parameter: n = Nummer des Joystickports (1 oder 2).

...komfortable Abfrage der aktuellen Stellung des Joysticks. Wenn er ruht, ist JOY (N) = 0. Bewegt man ihn in eine der acht möglichen Richtungen, bekommt JOY den entsprechenden Wert:

- 0: keine Aktion,
- 1: nach oben,
- 2: rechts oben,
- 3. rechts,
- 4: rechts unten,
- 5: nach unten,
- 6: links unten,
- 7: links,
- 8: links oben.

Wird zusätzlich der Feuerknopf gedrückt, erhöhen sich diese Zahlen um 128.

Damit läßt sich folgende Joystickabfrage in eigene Programme einbauen:

```
10 REM JOYSTICKABFRAGE PORT 2
20 FOR I=0 TO 7: READ J$(I): NEXT
30 FOR I=0 TO 7
40 IF JOY(2)=I THEN PRINT J$(I)
50 NEXT
60 GOTO 20
100 DATA "NICHTS","OBEN","OBEN RECHTS","RECHTS"
110 DATA "UNTEN RECHTS", "UNTEN", "UNTEN LINKS"
120 DATA "LINKS","OBEN LINKS"
```

```
**)**)F$ [vierstelliger Hexadezimalwert]**)**
```

...ersetzt dezimale Zahlenangaben: Ein Hexstring von $0000 bis $FFFF wird akzeptiert.

```
PRINT $C000
SYS $FCE2
```

% [achtstelliger Dualwert]

...kann ebenfalls statt Dezimalzahlen eingesetzt werden, allerdings nur von 0 bis 255.

```
PRINT %11110000
240
```

Achtung: Beide Funktionen (<$> und <%> vertragen keine Variablen, sondern nur konstante Werte!

Fehlermeldungen

In 99 von 100 Fällen werden Ihnen bei der Programmarbeit mit Special Basic die vertrauten Fehlermeldungen des Original-Basic 2.0 des C 64 unter die Finger kommen. Aufgrund der Fülle neuer Befehle mußten aber zwölf zusätzliche, numerierte Fehlerabfragen integriert werden:

- 31: UNTIL WITHOUT REPEAT

Sie verwenden im Basic-Programm die Anweisung UNTIL ohne einleitendes REPEAT (ähnlich NEXT WITHOUT FOR des Basic 2.0).

- 32: END LOOP WITHOUT LOOP

Der Basic-Interpreter trifft ohne vorhergehendes LOOP auf END LOOP.

- 33: MISSING END LOOP

...Sie haben vergessen, bei der entsprechenden Schleife END LOOP im Programm zu vermerken. Das Programm stolpert drüber, wenn es die Schleife mit EXIT verläßt.

- 34: MISSING BEND

...bei BELSE oder IF mit BEGIN wurde kein abschließendes BEND gefunden.

- 35: PROC NOT FOUND

Überprüfen Sie den exakten Namen der aufgerufenen Prozedur – falls er nicht ganz fehlt!

- 36: END PROC WITHOUT EXEC

Sie haben die Anweisung END PROC eingetragen, ohne per EXEC in ein Unterprogramm zu springen.

- 37: CANT RESUME

...s. Beschreibung zur Anweisung RESUME.

- 38: OUT OF RANGE

...wenn bei Befehlen zur Bildschirmgrafik das Rechteck (s. REC) größer als die Bildschirmbegrenzungen ist.

- 39: ILLEGAL STRING LENGTH

Diese Fehlermeldung kommt unweigerlich, wenn eine Zeichenkette die vorgegebene Ausdehnung überschreitet (länger als 255 Zeichen). Ebenso, wenn der Text zu KEY mehr als 31 Zeichen umfaßt oder die String-Länge von CHAR größer als 8 ist.

- 40: WRONG STRING

...gibt's nur dann, wenn Sie bei den Funktionen DEC, $ und % falsche String-Zeichen verwenden! Trägt man z.B. bei DEC eine Dezimalzahl ein, erscheint die Basic-2.0-Fehlermeldung TYPE MISMATCH.

- 41: MISSING ENDCASE

Sie haben vergessen, hinter einer CASE-OF-Struktur das abschließende ENDCASE im Basic-Programm einzutragen.

- 42: UNKNOWN VARIABLE ...vorher nicht als LOCAL definiert.

Sternenhimmel – Bildschirm als Teleskop

Tor zum Universum

Die elektronische Sternenkarte im Computer: Unser Programm zeigt auf Tastendruck, wo Planeten und Sternbilder am Nachthimmel stehen.

Für jeden beliebigen Zeitpunkt und jeden Standort der Erde kann man in Sekundenschnelle ein naturgetreues Abbild des Firmaments simulieren. Es enthält die bekanntesten Fixsterne, Sternbilder und Planeten, die auf dem nördlichen bzw. südlichen Sternenhimmel zu sehen sind.

Allerdings lassen sich unsere Nachbarplaneten im Sonnensystem nicht ohne weiteres mit bloßem Auge von x-beliebigen Fixsternen unterscheiden: Um sie zu lokalisieren, braucht man genaue Infos über den jeweiligen Standort. Diese Daten – auch über Sonne und Mond – sind im Programm integriert. Komplizierte Berechnungsroutinen garantieren stets den korrekten Himmelsausschnitt.

Laden und starten Sie die Erweiterung »Special Basic« (s. Beschreibung):

```
LOAD "SPECIAL BASIC",8
```

Aktivieren Sie das Grafiksystem mit RUN, dann starten Sie das Astronomie-Programm mit:

```
CHAIN "STERNENHIMMEL"
```

Nun fordert Sie die Special-Basic-Applikation auf, Uhrzeit (im Format HHMM, z.B. »2130« für halb zehn Uhr abends) und Datum einzugeben.

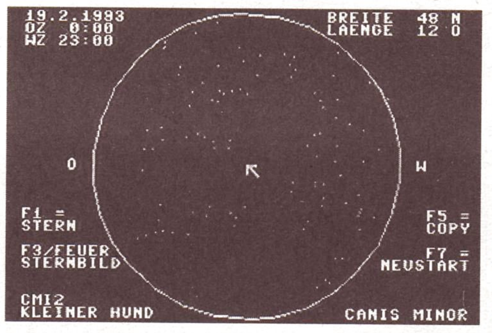
Anschließend bestimmt man den Standort des Betrachters: Werte zwischen 89,9 Grad nördlicher und südlicher Breite (– 89,9), bzw. dem 180sten westlichen und östlichen Längengrad (– 180) werden akzeptiert. Vorgegeben sind die Zahlen von Hamm/Westfalen. Andere Gradeinteilungen finden Sie in unserer Tabelle.

Wenn der Programminfo-Screen erscheint, ist eine Taste zu drücken – dann baut sich die Sternenkarte auf (Abb.). Zunächst zeichnet das Programm die Planeten, anschließend Pixel für Pixel die einzelnen Sternenbilder: Kleiner und Großer Wagen, Perseus, Pegasus, Zwillinge, usw. Wenn die Sternenkarte in voller Pracht erstrahlt, erscheint in der Bildschirmmitte der Mauszeiger. Er läßt sich mit dem Joystick in Port 2 oder per Cursor-Tasten steuern.

Die oberen Bildschirmzeilen geben Auskunft über die zuvor eingegebenen Werte. Außerhalb des Firmamentausschnitts sind die Funktionen des Bearbeitungsmenüs vermerkt, die man mit den entsprechenden Tasten aufruft:

<F1> **Stern:** Bewegen Sie den Pfeil in die Nähe des gewünschten Himmelskörpers im Kreisausschnitt. Per Joystick läßt sich dieser millimetergenau positionieren. Nach Tipp auf <F1> sucht der Computer den Stern und zeigt dessen Namen auf dem Bildschirm links unten. Mit <RETURN> kommt man erneut in den Such-Modus. Der Pfeil läßt sich jetzt wieder frei bewegen.

<F3>**/Feuer Sternbild:** In diesem Menüpunkt kann man die Sternensuche entweder per <F3> oder Druck auf den Feuerknopf aktivieren. Voraussetzung: Der Pfeil muß sich



Der Nachthimmel von München am 19. 2. 1993 um Mitternacht

»Sternenhimmel« (Koordinatenbeispiele)

Diese annähernden Breiten- und Längengradkoordinaten zeigen den Sternenhimmel wie er sich Bewohnern dieser Städte und Umgebung präsentiert.

Negative Vorzeichen der Gradangaben bedeuten:
- bei Breite: südlich des Äquators,
- bei Länge: östlich von Greenwich.

Ort	Breitengrad	Längengrad
Berlin	53	- 14
Bremen	53	- 9
Dresden	51	- 14
Frankfurt/Main	50	- 9
Hamburg	54	- 10
Hamm/Westfalen	52	- 8
Karlsruhe	49	- 9
Köln	51	- 7
Leipzig	52	- 12
Magdeburg	52	- 12
München	48	- 12
Rostock	54	- 12
London	52	0
Madrid	40	4
New York	40	70
Paris	49	- 2
Rio de Janeiro	- 20	40
Rom	42	- 13
San Francisco	35	123
Nordpol	90	0
Südpol	- 90	0

Die Koordinate 0,0 liegt im Atlantischen Ozean, ca. 550 Seemeilen südlich der Küste Ghanas (Afrika).

Kurzinfo: Sternenhimmel

Programmart: Grafikanwendung
Laden und starten: CHAIN "STERNENHIMMEL"
Besonderheiten: läßt sich per Tastatur oder Joystick Port 2 steuern
Benötigte Blocks: 59
Programmautor: Horst Hinkelmann

ebenfalls zumindest in der Nähe des gesuchten Sternbilds befinden. Die Reihe der gefundenen Himmelskörper beginnt zu blinken, per <RETURN> oder Tipp auf den Feuerknopf stellt man das ab und kann weitersuchen.

Sterne oder Bilder lassen sich auch per Tastatur suchen: Tippen Sie den gewünschten Namen »blind« ein (der erste Buchstabe erscheint im unteren Bildschirmfeld, jetzt kann man die fehlenden Zeichen dazutippen) – nach <RETURN> belegt der Pfeil automatisch die richtige Position.

<F5> **Copy:** ...aktiviert die Druckausgabe des aktuellen Grafikbildschirms. Der Drucker muß per seriellem Kabel bzw. Hardware-Interface mit dem C 64 verbunden sein. Denken Sie dran, das Gerät vorher einzuschalten, sonst bricht das Programm mit einer Fehlermeldung ab (»Device not present«)!

<F7> **Neustart:** ...unterbricht die aktuelle Sternenreise und kehrt zum Startbildschirm zurück. Dort können Sie Werte ändern und andere Betrachterstandorte bestimmen. Per <RUN/STOP RESTORE> läßt sich das Programm abbrechen.

Wer sich den Besuch des Planetariums ersparen will: Das Programm zeigt ebenso, wieviel Sternlein stehen... (bl)

Graphiccharts – Statistik zum Anfassen

Ein Bild sagt mehr als 1000 Zahlen

Nüchterne und knochentrockene Prozentwerte erwachen zum Leben: Unser Programm bringt sie blitzschnell als Balken-, Torten- oder Liniengrafik auf den Bildschirm.

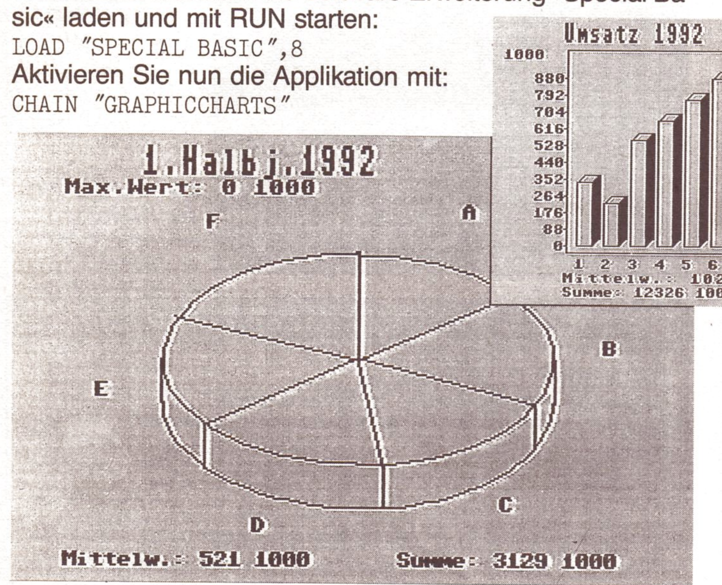
Wer gerne mit Statistiken jongliert und umfangreiche Listen mit Zahlen bzw. Prozentangaben zur Lieblingslektüre erkoren hat, kann eine Illustrierte daraus machen. Sechs verschiedene Grafikfunktionen lassen sich aufrufen, um abstrakten Zahlen professionellen Anstrich zu verleihen.

Zunächst muß man die Software-Erweiterung »Special Basic« laden und mit RUN starten:

```
LOAD "SPECIAL BASIC",8
```

Aktivieren Sie nun die Applikation mit:

```
CHAIN "GRAPHICCHARTS"
```



[1] Wer hat sich das größte Stück vom Kuchen abgeschnitten?

[2] Per Taste <F3> wählt man die 3D-Balkengrafik

Das Programm meldet sich mit dem Menübildschirm. Die einzelnen Punkte ruft man mit den revers angezeigten Tasten auf:

<I> **Infos zum Programm:** ...bringt wichtige Benutzerhinweise, z.B.:
- Hat sich eine Hires-Grafik aufgebaut, läßt sie sich nach Tipp auf <F3> als Hires-Bild auf Disk speichern (33 Blocks).
- Anschließend kann man z.B. eine Balkenstatistik mit Werteliste zum Drucker schicken, der über ein serielles Kabel oder entsprechendes Hardware-Interface mit dem C64 verbunden ist. Per Tastendruck geht's wieder zurück ins Hauptmenü.

<F1> **Balken variabel (1 bis 24):** ...präsentiert vorher eingegebene Zahlen in Balkenform. Zunächst fragt das Programm, ob man eine Zahlendatei (Endung: .dat) laden oder neue Werte eingeben möchte:

Neueingabe:
- Nennen Sie die Anzahl der gewünschten Werte – je nach Statistiktyp zwischen 1 und 24.
- Legen Sie die Überschrift (Titel) fest, die oberhalb des Hires-Screens eingeblendet wird (z.B. >Umsatz 1992>). Das Programm akzeptiert maximal zwölf Zeichen.
- Jetzt muß man den höchsten Wert eintragen, den ein Datenelement erreichen kann. Er darf nicht mehr als vier Stellen umfassen (also höchstens »9999«).
- Geben Sie die gewünschten Zahlenwerte ein (auf Nachkommastellen verzichten und darauf achten, daß die Zahl ebenfalls nicht länger als vier Ziffern ist!) und eine markante, zweistellige Bezeichnung (z.B. »Ja« oder »01« für Januar).
- Die bisherigen Eingaben kann man als sequentielle Datei auf Diskette sichern. Als File-Name muß der Titel zur Grafik herhalten (deshalb nur zwölf Zeichen, ergänzt mit der Kennung ».dat«).
- Auf Wunsch lassen sich Grafikausgabe und Listendruck mit zusätzlichen Infos ausstatten: Mittelwert, Gesamtsumme, Prozentberechung und Kommentar (erscheint nur beim Ausdruck).
- Zuletzt bestätigt man mit <J>, daß der angeschlossene Drucker eingeschaltet ist. Anschließend baut sich die Balkenstatistik auf. Per <F3> speichert man die Hires-Grafik nach Eingabe eines Dateinamens auf Diskette, <F1> leitet die Druckausgabe ein. Dann kehrt das Programm wieder ins Hauptmenü zurück.

Datei von Diskette laden:

Wer Zahleneingaben sichert, kann sie später beliebig oft in Form jedes gewünschten Statistiktyps auf den Bildschirm bringen. Dazu gibt man den maximal zwölf Zeichen langen Dateinamen ein, drückt <RETURN> und beantwortet alle weiteren Computerfragen (»Mittelwert (j/n)« usw.). Probieren Sie's mit unserer Demodatei »Umsatz 1992« auf Diskette aus!

Weitere Statistiktypen des Programms:
- <F2> Linien variabel (1 bis 24 Wertangaben),

Kurzinfo: Graphiccharts

Programmart: Grafikanwendung
Laden: CHAIN "GRAPHICCHARTS"
Starten: automatisch nach dem Laden
Besonderheiten: Grafikdaten lassen sich wahlweise als SEQ-Datei oder Hires-Bild speichern!
Benötigte Blocks: 46
Programmautor: S.I.G/H. Beiler

- <F3> Balken 3D variabel (1 bis 12),
- <F4> Tortengrafik (1 bis 6), s. Abb. 1,
- <F5> Pareto % variabel (1 bis 12),
- <F7> Münzen variabel (1 bis 6).

Die Dateneingabe und daraus resultierende Programmabfragen entsprechen exakt Menüpunkt <F1>.

<F6> **Directory:** ...bringt das Disketteninhaltsverzeichnis der aktuellen Scheibe im Laufwerk,

<E> **Ende:** ...steigt ohne Reset aus und aktiviert das Basic 2.0.

Gespeicherte Hires-Grafiken haben 36 Blocks auf Diskette, besitzen aber $00 als Startadresse und einen modifizierten Byte-Aufbau, den nur Special-Basic versteht. So holen Sie Hires-Bilder wieder auf den Bildschirm:

```
10 HIRES 0,1
20 GLOAD "(Dateiname der Grafik)"
30 GETKEYA$: NRM
```

Sorry, mit Hi-Eddi (oder ähnlicher Software) läßt sich dieses Grafikformat nicht nutzen, auch wenn man die Ladeadresse vorher z.B. auf $2000 setzt. Verwenden Sie dazu das Utility »Special-Saver« (s. Rubrik »Tips und Tricks«!) (bl)

Random-Copy – von Bild zu Bild

Sanfter Schnitt

Kennen Sie den sanften Übergang von einer Filmszene zur nächsten? Das kann der C 64 auch!

Sicher stört es Sie auch, wenn beim Anzeigen mehrerer Grafiken auf dem Bildschirm der Wechsel sehr abrupt erfolgt, wünschen sich dann lieber einen fließenden Übergang, ähnlich dem Umblenden im Kinofilm. Das Programm »Random Copy« bringt dem Tausendsassa C64 genau dies bei.

In den Abb. 1 bis 8 sehen Sie die einzelnen Phasen dieses Übergangs, von den ersten sichtbaren Punkten zu Beginn über das erste komplette Bild bis zum Erscheinen des nächsten.

Sie können das Programm mit

```
LOAD "RANDOM COPY",8,1
```

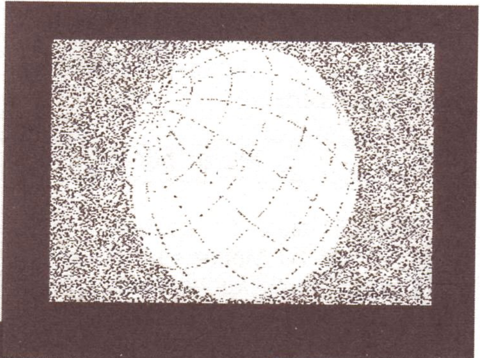
in den Computer laden. Der Start erfolgt mit

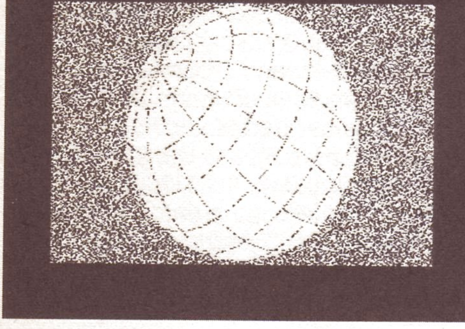
```
SYS 828
```

Dazu muß in Speicherstelle $03F8 (dez. 1016) das höherwertige Byte der Grafikstartadresse stehen. Beispiel: Beginnt das Bild bei $2000, müssen Sie mit

```
POKE 1016,32
```

die Adresse übergeben. In Speicherstelle $03f9 (dez. 1017) braucht das Programm das höherwertige Byte der Zieladresse, also des aktuellen Videospeichers. Ist dort schon eine Grafik, die gerade angezeigt wird, vorhanden, wird sie allmählich durch die neue überschrieben. War der Bereich leer, wird das neue Bild sanft eingeblendet.





[1] Zu Beginn sind da nur wenige Punkte...

Random Copy übernimmt nicht die Anzeige der Grafik, sondern nur den Datentransport innerhalb des Speichers. Für Anzeige und Laden von Diskette muß der Benutzer selbst sorgen, indem er die entsprechenden Befehle in sein Hauptprogramm einbaut.

Übrigens kann das Programm auch das RAM unter dem Betriebssystem ($A000 bzw. $E000) benutzen und arbeitet daher mit Simons Basic zusammen. Außerdem liegt es so im Speicher, daß kein Basic-Speicher verbraucht wird. Beachten Sie auch, daß Random Copy nicht zusammen mit dem Floppyspeeder Jiffy-DOS läuft. (hb)

[2] ...die sich bald zu einem Bild formen...

[3] ...bis das nächste erscheint

Kurzinfo: Random Copy

Programmart: Grafik-Tool
Laden: LOAD "RANDOM COPY",8,1
Start: SYS 828
Besonderheiten: zeigt Grafik nicht selbst an
Benötigte Blocks: 1
Programmautor: Stephan Arndt

Bit, Byte, Sprite & Co.

Um hochauflösende Grafik oder Sprites auf den Bildschirm des C 64 zu bringen, muß man sich in der Speicherverwaltung des Video-Interface-Controllers (VIC) gut auskennen. Hier finden Sie jede Menge Routinen und Utilities, die Ihnen manche Programmierarbeit abnehmen.

Calcul-Ace: Taschenrechner im Hires-Format

Billige Minicomputer für die Westentasche gibt's wie Sand am Meer – oftmals kosten sie nicht mal zehn Mark. Unser grafischer Taschenrechner aber ist kostenlos:

```
LOAD "CALCUL-ACE",8
```

Um korrekt zu funktionieren, lädt er nach dem Start mit RUN vier Dateien nach:

- Graphix ($5E00 bis $77F): die Benutzeroberfläche der Taschenrechnersimulation, die inkl. geändertem Zeichensatz auf den Screen geplottet wird,
- Bas ($0801 bis $2EA5): das Steuerprogramm in Basic,
- Allsprites (ab $E400): ...enthält die Muster der im Programm verwendeten Sprites,
- IRQ.Obj ($C000 bis $C57F): Assembler-Teil, der den Interrupt auf $C093 verbiegt.

[1] Calcul-Ace: gelungene Taschenrechner-Simulation mit dem C 64

Anschließend steht der Taschenrechner mit den üblichen Funktionen bereit (Abb. 1). Man steuert ihn per Joystick Port 2. Ein Tipp auf den Feuerknopf aktiviert die gewünschte Rechenfunktion (hyp, sin, cos, tan usw.). Das nach Einschalten von INV ON erscheinende Menü bietet nützliche Umrechnungsarten (Dezimal in Hexadezimal, Binär in Dezimal usw.). Der Knopf CE (unten rechts) löscht das Eingabefeld. Das Programm läßt sich nicht mit der Tastenkombination <RUN/STOP RESTORE> abbrechen – Sie müssen den Resetknopf drücken oder den Computer ausschalten.

(Norbert Nagy/bl)

Super-Cursor in neuem Gewand

Es gibt nichts langweiligeres als inverse Cursor-Blöcke, die auf dem Bildschirm stur vor sich hinblinken. Unser Programm ersetzt den stupiden Normal-Cursor durch ein sternförmiges Sprite, das dem Screen eine spritzige Note verleiht:

```
LOAD "SUPER-CURSOR",8
```

Nach dem Start mit RUN blendet sich der Bildschirm während des Entpackens kurz aus und meldet sich mit dem neugestalteten Cursor wieder. An der – Programmierern vertrauten – Speicherkonfiguration hat sich nichts geändert: Der Basic-Anfang liegt bei $0801 (2049), das Ende des Basic-RAM bei $9FFF (40959). Die Cursor-Geschwindigkeit läßt sich ändern:

```
POKE 56325,255
```

Nach dem Laden belegt das Programm den Speicherbereich von $0801 bis $0DA1, mit RUN verschiebt es die Daten in Adressen ab $C33C (49980). Dort wird der Systeminterrupt auf eine Routine ab $C100 (49408) gelenkt, die den neuen Cursor jede $^1/_{60}$stel Sekunde anzeigt: Die nötigen Sprite-Daten liegen davor (von $C000 bis $C0FF). Falls man mit <RUN/STOP RESTORE> aussteigt, läßt sich der Cursor im neuen Gewand per SYS 49980 erneut initialisieren. Wer den Super-Cursor in eigene Programmentwicklungen einbauen möchte, muß den relevanten Speicherbereich per geeignetem Maschinensprache-Monitor (z.B. SMON ab $8000) sichern:

```
S "SUPERCURSOR" 08 C000 CA00
```

Nach dem absoluten Laden (per »,8,1«) innerhalb eines Basic-Programms wird das Cursor-Sprite ebenfalls mit SYS 49980 initialisiert.

(Tom Töper/bl)

Immer nur lächeln...

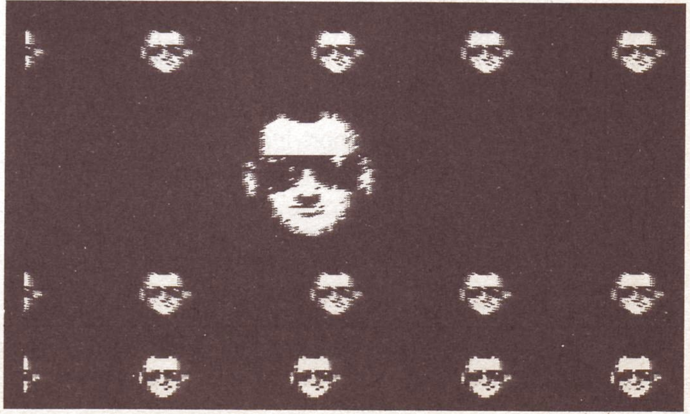
Mehr als 20 Mafiosi-Fratzen auf ein- und demselben Bildschirm? Das hört sich gefährlicher an, als es ist:

```
LOAD "SMILE",8
```

Nach RUN gleiten per geschickter Rasterzeilen-Interrupt-Programmierung 20 sonnenbebrillte Gaunervisagen auf dem Screen hin und her. Der Oberschurke in der Mitte besteht aus zwei in x- und y-Richtung gedehnten Sprites, die sich während der Scroll-Bewegung ständig überlagern. Dadurch entsteht der Eindruck, als grinse der Kerl hinterhältig (Abb. 2).

Unmittelbar nach dem Laden belegt das Demo-Programm den Basic-Speicher von $0801 (2049) bis $126B (4715). Die beiden Sprite-Muster sind ebenfalls darin enthalten. Beachten Sie zur Technik der Sprite-Überlagerung unseren Grundlagenartikel (Rubrik »Sprites«) und das Demo-Programm »Anima+Overlay« auf der Diskette zu diesem Sonderheft!

(H.W. Müller/bl)



[2] Smile: Wer zuletzt lacht, lacht am besten...



Grafik in versteckten RAM-Speichern

Für den C64 existieren zwei System-Erweiterungen, die mit leicht verständlichen Befehlen vor allem Hires-Grafik unterstützen: das legendäre Simon's Basic (in manchen Fachgeschäften noch als Modul für den Expansion-Port erhältlich) und Special Basic (s. Beschreibung in diesem Heft). Funktionen zum Speichern hochauflösender Grafik fehlen bei Simon's Basic völlig und wurden auch in Special Basic nur unzureichend verwirklicht: Das Programm verwendet ein spezielles Format zur Datenablage (Anweisung GSAVE). Solche Bilder lassen sich nur mit GLOAD wieder in den Speicher holen. Außerdem benutzen beide Basic-Erweiterungen einen schwer zugänglichen Speicherbereich für die Grafikbytes: das RAM unterm ROM von $E000 (57344) bis $FFFF (65535).

Sie merken es spätestens dann, wenn Sie versuchen, die 8192 Byte große Grafikdatei mit BSAVE (statt GSAVE) zu sichern: Sie erwischen stets Inhalte der ROM-Adressen (Betriebssystem-Kernel), die verständlicherweise mit Grafik nichts zu tun haben. Unsere beiden Utilities machen's anders:

Special-Saver: ... ist eine Maschinensprache-Routine, die man mit beiden Erweiterungen (Simon's oder Special Basic) einsetzen kann. Laden und aktivieren Sie das Grafiksystem, mit dem Sie arbeiten, dann das Utility:

```
LOAD "SPECIAL-SAVER",8,1
```

Das Programm nistet sich bei beiden Basic-Erweiterungen im freien RAM-Bereich von $7F00 (32512) bis $7FD1 (32721) ein. Initialisiert wird es mit:

```
SYS 32512
NEW
```

Erzeugen Sie nun mit den entsprechenden Befehlen eine Hires-Grafik (einfarbig oder bunt). Durch die Anweisung:

```
SYS 32521
```

(am besten im Programm-Modus nach Tastendruck):

```
10 HIRES 0,15
20 Grafikanweisungen
1000 POKE 198,0: WAIT 198,1
1010 SYS 32521
```

sichert man das Bild als Datei im Hi-Eddi-Format mit der üblichen Startadresse $2000 (8192). Als File-Name verwendet das Programm automatisch »SimonPic.xx« (xx = zweistellige Zahl, bei »00« beginnend). Achtung: Auf der Disk müssen unbedingt 37 Blöcke frei sein, da der Inhalt des Farb-RAM automatisch mitgespeichert wird! Selbstverständlich sollte man die Bilddateien nach dem Speichern umbenennen und mit markanten Namen versehen. Ab sofort lassen sich solche Grafiken von anderen C-64-Zeichenprogrammen (z.B. Starpainter, Art Studio usw.) laden und weiterverarbeiten.

Special-Loader: ... macht das Gegenteil: Es lädt Grafiken im Hi-Eddi-Format von Diskette und transportiert sie in den bei Simon's- und Special Basic vorgesehenen Speicherbereich ab $E000 (57344). Lediglich die Lage des Farb-RAM im Speicher stimmt nicht überein: Hi-Eddi verwendet 1000 Byte ab $4000 (16384), Simon's Basic ab $C000 (49152) und Special Basic ab $CC00 (52224). Darauf richtet das Maschinenprogramm HE/SB ein (es belegt nach dem Laden den Kassettenpuffer im Bereich von $033C (828) bis $038B (907), kann aber in jeden anderen freien Speicherbereich des C64 verlegt werden):

```
LOAD "HE/SB",8,1
```

Zwei SYS-Anweisungen stehen zur Verfügung:

- SYS 828: lädt farbige Hi-Eddi-Hires-Bilder (37 Blocks) von Diskette,
- SYS 867: einfarbig. Da man hier das Farb-RAM nicht berücksichtigt, sollten Sie vorher hochauflösende Bildschirme per HIRES-Befehl mit den gewünschten Farben belegen.

Zunächst muß man die Grafik im Hi-Eddi-Format in den Speicher (»,8,1«) holen oder den Ladebefehl in ein Basic-Programm einbauen – wie in unserem Demo-Programm, das Sie ebenfalls wahlweise mit Simon's- oder Special Basic verwenden können:

```
LOAD "SPECIAL-LOADER",8
```

Achtung: Das Listing sollten Sie nicht ändern (Zeilen löschen oder hinzufügen), denn es modifiziert sich selbst! Das ist nötig, um die unterschiedliche Lage der Farb-RAMs beider Basic-Erweiterungen unter einen Hut zu bringen. Außerdem ist das Programm auf die Farbgrafik MENUE zugeschnitten, die wir leider aus Platzgründen nicht mehr auf der beiliegenden Sonderheft-Diskette unterbringen konnten. Sie befindet sich auf der Disk zum 64'er-Sonderheft 75.

Es dürfte aber keine Probleme bereiten, mit zwei, drei Listingzeilen und eingebauter INPUT-Abfrage eine eigene Programmroutine zum Laden Hi-Eddi-Grafiken für Simon's- oder Special Basic zu entwickeln (s. Demo-Listing HI-EDDI/SIMON auf beiliegender Disk!).

Eine Änderung müssen Sie allerdings beachten, wenn Sie HE/SB als Ladehilfe ausschließlich mit Special Basic einsetzen möchten: Nach dem Laden muß man mit POKE 839,204 die Maschinenroutine aufs Hires-Farb-RAM richten, das Special Basic verwendet. Wer einen Maschinensprache-Monitor besitzt, sollte den Befehl ab Adresse $0346 ändern:

```
A 0346 LDA #$CC
```

und die Assemblerroutine erneut speichern:

```
S "HE/SPECBAS" 08 033C 038B
```

Feinschliff für Koala-Painter-Bilder

Neben Amica Paint und Paint Magic gehört das Multicolor-Malprogramm Koala-Painter zur Standardausrüstung eines jeden C-64-Grafikers.

Unser Utility bearbeitet Farbbilder dieser Grafikanwendung, die stets mit dem charakteristischem Prefix »Apic« beginnen:

```
LOAD "KOALAMODIFY,8
```



[3] Koalamodify: Grafikumrisse glätten mit dem Optimize-Befehl

Nach dem Start mit RUN fragt das Programm nach dem Namen des gewünschten Koala-Bildes. Es genügt, wenn man zwei oder drei Zeichen des gesuchten File-Namens eingibt – allerdings ohne Prefix und Kennbuchstaben, sonst erzeugen Sie eine Fehlermeldung! Das Programm verwendet automatisch den Joker <*> zum Laden der gewünschten Datei. Unsere beiden Koala-Grafiken auf Diskette (Apic p wern und Apic c koala) lädt man z.B. mit »WERN« oder »KOA*«. Wenn die Floppy ihre Arbeit erledigt hat, erscheint das Menü. Die Programmfunktion werden per entsprechender Taste aktiviert:

<SPACE> Menu oder Picture: Wechselweise taucht bei Tipp auf <SPACE> die Multicolorgrafik oder das Menü auf.

<B> Change Backgroundcolor: ...ändert alle Grafikflächen, die mit der gespeicherten Hintergrundfarbe belegt sind. Erneuter Druck auf die SPACE-Taste zeigt die Änderungen.

<I> Invert Picture: ...EOR-Verknüpfung aller Grafikfarben,
<X> Horizontal Flip: ...dreht die Grafik in x-Richtung (quasi als Spiegelbild),
<Y> Vertical Flip: ...stellt das Bild auf den Kopf,
<O> Optimize Bits: ...ist eine nützliche Funktion des Utilities: Das Bild wird Pixel für Pixel gescannt und nachbearbeitet. Kantige Umrisse und eckige Konturen (ein gravierender Nachteil von Multicolorgrafiken) verlaufen jetzt feiner und glatter (Abb. 3).

Cursors Move Picture: Mit den Cursor-Tasten läßt sich das Bild in vier Richtungen verschieben: nach oben, unten, links und rechts. Wenn Grafikpixel am Rand verschwinden, erscheinen Sie auf der anderen Seite wieder im Bild.

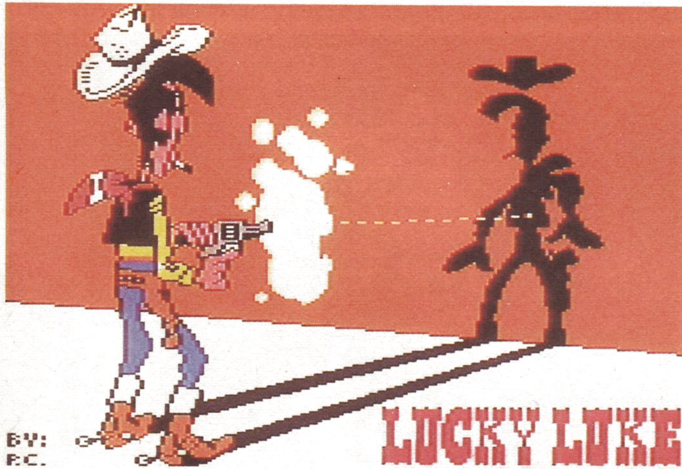
<S> Save Picture: ...ist die wichtigste Programmfunktion: Geänderte, optimierte, gespiegelte oder verschobene Grafiken kann man in der neuen Einstellung mit anderem Dateinamen wieder auf Diskette speichern.

Falls Sie andere Multicolorgrafiken verwenden möchten (z.B. Amica Paint oder Paint Magic), muß man sie zuerst mit einem entsprechenden Programm oder Modul konvertieren (z.B. Slide-Konvert oder Action Replay). (Matthias Kranz/bl)

Grafik-Entzerrer

Unsere Diashow im 64'er-Sonderheft 63 bringt tolle Multicolorgrafiken auf den Screen. In eigene Programme konnte man diese Grafiken aber bisher nie einbauen: Die Bilder auf Diskette sind nämlich gepackt!

»Slide-Konvert« schüttelt solche Grafiken kräftig durch, bis sie wieder als Koala-Painter-Bild im Speicher stehen. In diesem Format lassen sie sich speichern:



[4] Slide-Konvert: macht aus Diashow-Grafiken Koala-Bilder

```
LOAD "SLIDE-KONVERT",8
```

Hat man mit RUN den Startschuß abgefeuert, sollten Sie die Disk von Sonderheft 63 ins Laufwerk legen und die gewünschte Bildnummer (01 bis 19) angeben. Das einleitende Grafikzeichen, mit dem jedes Diashow-Bild ausgestattet ist, wird vom Programm automatisch gesetzt.

Wenn die Grafik geladen ist, wird sie entpackt und läßt sich per <SPACE> in aller Ruhe betrachten oder sofort mit <RETURN> als Koala-Painter-Grafik auf Diskette speichern (mindestens 40 Blöcke müssen noch frei sein!). Wer die Sonderheft-Disk Nr. 63 nicht besitzt, kann mit der Grafik »08 LUCKY LUKE« auf der beiliegenden Disk zu diesem Sonderheft einen »Versuch am lebenden Objekt« starten (Abb. 4)!

(Matthias Kranz/bl)

Sprites und Grafik in fremden Programmen aufspüren

Es gibt haufenweise tolle Action-Games und Grafikadventures mit Super-Hires-Bildschirmen oder akribisch entworfenen Sprites, bei denen jedes Pixel am rechten Platz ist. Nach dem Griff zum Resetknopf ist's zwar vorbei mit der Herrlichkeit, meist sind aber die Hires-Screens und Sprite-Muster in den Speichertiefen des C 64 noch unversehrt erhalten geblieben.

Unser Grafikspürhund durchkämmt den Speicher:

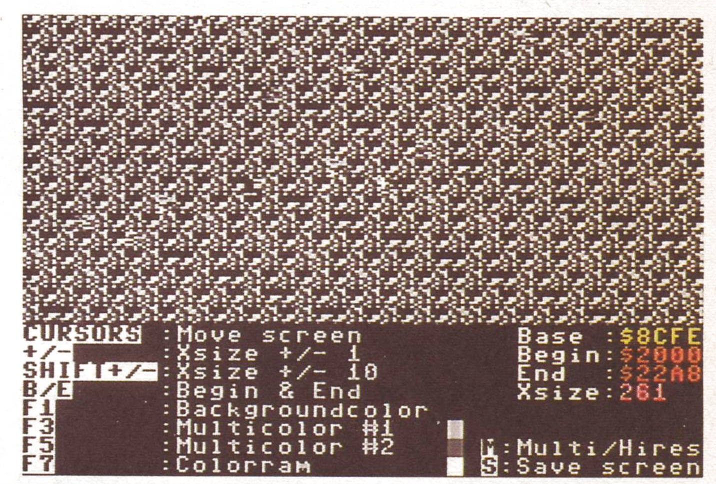
```
LOAD "GFXRIP $1000",8,1
```

Starten Sie das Utility mit SYS 4096.

Das Hauptmenü bietet fünf Optionen (Abb. 5):

<F1> Ripp Charset: ...sucht nach versteckten Zeichensätzen,
<F3> Ripp Screens: ...macht per geändertem Zeichensatz erzeugte Grafikbildschirme ausfindig (Achtung: keine Hires-Screens!),
<F5> Ripp Sprites: ...entdeckt verborgene Spritemuster,
<←>: Damit kommt man aus jedem Untermenüpunkt wieder ins Hauptmenü,
<RUN/STOP> Exit: ...schaltet den Grafik-Grabber ab. Mit SYS 4096 wird das Programm erneut aktiviert.

Nach Wahl jedes Hauptmenüpunkts taucht ein entsprechendes Untermenü mit dem Scanner-Bildschirm auf. Die Funktionen sind im Prinzip identisch und selbsterklärend:
- <+>, <->: im Speicher vor- oder rückwärts blättern. Im Scanner-Bildschirmausschnitt kann man mitverfolgen, wie sich die Byte-Muster verändern.
- <M>: Multicolormodus ein/aus,
- <F1> bis <F7>: Bildschirmfarben ändern,
- <S>: gewünschten Scanner-Ausschnitt speichern (Zeichensätze, Hires-Screens und Sprites). Die Datei »Probe-Sprite« enthält den Speicherbereich ab $6C00 nach dem Sichern auf Diskette, der allerdings nur aus zufälligen Sprite-



[5] GFXRip $1000: Alle Funktionen aktiviert man im Hauptmenü

Mustern besteht. Um das Utility zu testen, sollten Sie besser ein Super-Game laden (z.B. Turrican II), es mit Reset abbrechen und GFXRip laden. Dann kann man den Speicher nach den Spielegrafiken, geänderten Zeichensätzen und tollen Sprite-Mustern durchsuchen. (Matthias Kranz/bl)

Stamp-Packer: Grafik in Briefmarkengröße

Wers 64'er-Sonderheft 77 besitzt, kennt das Programm »Stamp-Maker V2«: Es erzeugt und verwaltet Grafikausschnitte (Stamps) von »Amica Paint«-Bildern. Damit kann man z.B. Animationen zusammenstellen, ohne stets



auf Hires-Bilder in voller Größe zugreifen zu müssen (ein einziger Hires-Screen belegt immerhin 8 KByte Speicher des C64!). Für alle, denen solche Teilgrafiken noch immer zu viel Speicherplatz verbrauchen, gibt's nun den »Stamp-Packer V1«, der die Grafikausschnitte nochmals staucht. Selbstverständlich sind diese Mini-Grafiken im Briefmarkenformat völlig kompatibel zu Stamp-Maker.

Laden Sie das Utility von der beiliegenden Diskette mit:

```
LOAD "STAMP-PACKER V1",8
```

Nach dem Start mit RUN holt sich der Computer die Maschinenprogramme »Stamp-Maker V2« ($C000 bis $C3EB) und »Stamp-Repack« ($C400 bis $C49B) in den Speicher.

Dann fordert Sie das Progamm auf, die Diskette mit der Amica-Paint-Bildersammlung einzulegen und den gewünschten Dateinamen anzugeben: Die charakteristische Datentypbezeichnung [B] ist unbedingt wegzulassen! Anschließend trägt man die Stamp-Ausdehnung in x- und y-Richtung ein (x = maximal 31, y sollte nicht größer als x sein, Abb. 6). Der Adreßbereich von 24576 bis 34577 wird mit Null-Bytes aufgefüllt.

Wenn der Grafikausschnitt erscheint, müssen Sie <SPACE> drücken und eine Diskette mit mindestens 40 freien Blöcken einlegen. Erneuter Tastendruck sichert die Mini-Grafik als Koala-Painter-Bild auf Disk (Ladeadresse: $6000). Laden Sie nun diese Koala-Grafik mit Amica-Paint und speichern Sie es in diesem Modus.

Tabelle. Die Parameter bestimmen den Grafikausschnitt

Stamp-Packer V1 (Parameterliste)

Variable	Funktion
x1	x-Ausdehnung des Grafikausschnitts
y1	...in y-Richtung
b1	Startadresse der zu lesenden Bitmap
z1	...des Video-RAM
c1	...des Color-RAM
b2	Startadresse des zu schreibenden Grafikbereichs
z2	...des Video-RAM
c2	...des Farb-RAM
ro	Bereich (0 = normales RAM, 1 = RAM unterm ROM)
pa	...definiert, ob man den Bereich lesen oder schreiben will (1 = Stamp aus Bitmap konvertieren, 0 = Stamp in Bitmap-Format wandeln und schreiben). Ist pa = 1, werden die Bytes ab b1, z1 und c1 aus der Bitmap in ein Stamp verwandelt und nach b2, z2, c2 geschrieben. Falls pa = 0 ist, läuft's umgekehrt: aus einem Stamp wird wieder eine Bitmap. Um den Bereich auszuschneiden, müssen die Parameter b1, z1 und c1 die Startadressen des Bildes enthalten; b2, z2 und c2 die des Speicherbereichs, in dem Sie die Mini-Grafik ablegen möchten (pa = 1). Will man das Stamp lesen und auf eine Grafikseite setzen, muß pa den Wert 0 haben. Die Variablen b2, z1 und c1 enthalten dann die Startadressen der Grafikseite.
x2	x-Position der Grafikseite,
y2	y-Koordinate

In 64'er-Sonderheft 77 finden Sie die vollständige Beschreibung zu »Stamp-Maker V2«.

```
STAMP-PACKER V1
WRITTEN 1992 BY JOERG NEUMANN

NAME AMICA-BILD ? SKAT
CURSORPUNKTE X  ? 20
CURSORPUNKTE Y  ? 20

BILD-DISK EINLEGEN!
```

[6] Stamp-Packer: Achten Sie auf fürs Programm akzeptable x- und y-Werte!

Das ist das Stichwort fürs nächste Utility von Stamp-Packer:

```
LOAD "PACK-PART 2",8
```

Nach RUN wird der Maschinenspracheteil »Zero-Kill« ($C500 bis $C555) nachgeladen, der alle vorher integrierten Null-Bytes wieder entfernt. Das macht nämlich Amica-Paint automatisch, da die Stamp-Grafik nicht die verlangten 10000 Byte eines Koala-Painter-Bildes belegt. Die gepackte Grafik wird auf Diskette verewigt, wenn man Namen und Zieladresse eingegeben hat.

Zum Entpacken braucht man die Routinen Stamp-Maker V2 und Stamp-Repack, die man per Endung »,8,1« in den Speicher holt. Mit der Eingabe von

```
SYS 50176, Startadresse Stamp
```

läßt sich die Mini-Grafik wieder entzerren.

Stamp-Repack liest die Daten ab Startadresse und schreibt den entpackten Code ins RAM unterm ROM ab $D000. Anschließend nistet sich das entpackte Bild wieder ab derselben Startadresse ein. Vorteil: Es wird kein zusätzlicher Speicher benötigt.

Jetzt kann man den Stamp normal anzeigen:

```
SYS 49152,x1,y1,b1,z1,c1,b2,z2,c2,ro,pa,x2,y2
```

Die nächste Befehlseingabe erledigt das Entpacken und die Bildschirmanzeige in einem Aufwasch:

```
SYS 50323, Startadresse Stamp, x1,y1,b1,z1,c1,b2,
z2,c2,ro,pa,x2,y2
```

Achten Sie darauf, daß pa = 0 ist. Die Bedeutung der Parameter finden Sie in unserer Tabelle.

Beachten Sie die Grafik [B]Skat auf unserer Sonderheftdiskette, mit der man die Funktionen ausprobieren kann.

(Jörg Neumann/bl)

Flackernde Rasterzeilen

Wer oft mit dem Rasterzeilen-Interrupt arbeitet, hat bestimmt festgestellt, daß manche Zeilen flackern. Der Grund: Bei einigen Rasterzeilen braucht der VIC-Chip länger als bei anderen, da er die Daten für die neue Bildschirmzeile aufbereiten muß. Diese Stolpersteine findet man ab Rasterzeile 51 in Abständen von jeweils acht Zeilen. Verwenden Sie folgende Berechnungsformel:

```
z = 51 + n * 8
```

N ist eine Variable mit einem Wert zwischen 0 und 30 (damit erfaßt man auch den Bildschirmbereich außerhalb des Textfeldes), das Ergebnis Z repräsentiert dann die Rasterzeilennummern, die man tunlichst meiden sollte.

(M. Kühlewein/M. Koch/bl)

Wellenbad

Das folgende Listing läßt den Bildschirm erzittern: In rhythmischen Zyklen erzeugt es Wellenbewegungen. Damit kann man z.B. Explosions-Gags in eigenen Adventures realisieren oder den Screen am Ende eines verlorenen Games zum Vibrieren bringen:

```
90 REM WAVE-SCREEN
100 S=0: FOR A=832 TO 894: READ B: S=S+B
110 POKE A,B: NEXT
120 IF S<>8143 THEN PRINT "FEHLER IN DATAS":END
130 PRINT"INITIALISIERUNG O.K.
140 DATA 32,241,183,142,121,3,169,0
150 DATA 133,198,173,22,208,72,169,0
160 DATA 141,22,208,162,7,238,22,208
170 DATA 32,120,3,202,208,247,162,7
```

```
180 DATA 206,22,208,32,120,3,202,208
190 DATA 247,165,198,201,0,240,228,104
200 DATA 141,22,208,169,0,133,198,96
210 DATA 160,10,200,208,253,96,10
```

Das Bildschirmbeben erzeugt man mit der Direkteingabe:

```
SYS 832,<Wert>
```

Der Parameter »Wert« muß eine Zahl zwischen 0 (niedrigste) und 255 (höchste Stufe) sein: die Frequenz des Wackel-Effekts. Per Tipp auf eine beliebige Taste brechen Sie das Programm ab.

»Wave-Screen« läßt sich auch in eigenen Basic-Programmen aufrufen. Möchte man die Mini-Erweiterung in Assembler-Routinen einbauen, muß man den Frequenzwert in Adresse $0379 (889) ablegen. Statt nach $0340 (832) ist dann nach $0346 (838 zu verzweigen: Damit umgeht man die Abfrage des Parameters »Wert«.

Schnelle Sprite-Animation

Unser Listing zeigt den Weg, wie man in Basic 2.0 Sprites kontinuierlich bewegt – ohne Assembler-Routine! Das jeweilige Sprite-Muster wird zur Demonstration aus dem Bereich von $A000 bis $BFFF in 64-Byte-Schritten geholt und in Adresse 704 (Sprite-Block 11) transferiert. Wer andere Speicherbereiche mit echten Sprite-Mustern auf den Bildschirm bringen will, muß die Zeilen 20 (BG=40960) und 90 (den Wert 49151) entsprechend anpassen:

```
10 VIC=53248: SPR=11: POKE 2040,SPR
20 I=SPR*64: BG=40960: AN=63
30 POKE VIC+23,1: POKE VIC+29,1
40 POKE VIC,24: POKE VIC+1,50: POKE VIC+39,0
50 PRINTCHR$(147)CHR$(14)"SPRITE-ANIMATION"
80 POKE VIC+21,1
90 FOR I=BG TO 49151 STEP AN: QU=I GOSUB 2100
110 NEXT
150 POKE VIC+21,0
998 END
999 :
2100 Z=ZI: GOSUB 2200: POKE 53,LO: POKE 54,HI
2105 Z=QU: GOSUB 2200: POKE 781,LO: POKE 782,HI
2110 POKE 780,AN+1: SYS 46728
2200 HI=INT(Z/256): LO=Z-HI*256
2210 RETURN
```

Das Basic-Programm benutzt dabei die Systemroutine zur String-Speicherung ab $B688. (bl)

Sprite-Kollisionen

Was wäre ein Programm mit quirligen Sprites ohne Reaktion auf Zusammenstöße, z.B. mit der Bildschirm-Hintergrundgrafik? Öde und langweilig!

Dabei lassen sich solche Kollisionen unkompliziert feststellen. Ein bißchen schwierig wird's aber, wenn Zusammenstöße nur bei bestimmten Bildschirmzeichen registriert werden sollen.

Zunächst muß man den Inhalt des VIC-Registers $DO1F (53279) lesen: Es reagiert auf Sprite-Kollisionen mit dem Screen-Hintergrund. Zu jedem der acht möglichen Sprites gehört ein Bit. Ist der Zusammenstoß passiert, muß man sofort die Position des beteiligten Sprite festhalten (gespeichert in den Registeradressen $D000 bis $D00F). Die Koordinaten von Sprite 1 (#0) sind also in $D000 (x-Richtung) und $D001 (y-wert) abgelegt.

Übersteigt der x-Wert die Zahl 255, muß man zusätzlich Adresse $D010 (53264) berücksichtigen:

```
SY = PEEK(53249)
SX = PEEK(53248)+256*(PEEK(53264) AND 1)
```

Die Variablen sx/sy bestimmen den Ort des Ereignisses für Sprite 1. Allerdings ergibt sich das Problem, daß die Sprite-Koordinaten nicht mit denen des Bildschirmzeichens übereinstimmen. Die müssen durch weitere Rechenformeln geliefert werden (Spalte und Zeile als Textbildschirm-Positionen):

```
S = INT(40*(SX-24)/320)
Z = INT(25*(SY-50)/200)
```

Aus dieser Ortsangabe berechnet man die Bildschirmspeicherstelle, die den Code des entsprechenden Zeichens enthält (s. Handbuch des C64). Jede Bildschirmzeile belegt im Speicher 40 Byte:

```
40 * (Z-1)+S
```

Addieren Sie das Ergebnis zur Startdadresse des Bildschirm-RAM (z.B. 1024 minus 1, da die erste Position schon in der Startadresse enthalten ist). Den Zeichencode erhält man mit:

```
C = PEEK(1023+40*(Z-1)+S)
```

Anschließend ist zur Routine zu verzweigen, die das Programm auf die Sprite-Kollision reagieren läßt.

(Heimo Ponnath/bl)

Sprites sauber ausblenden

Hat man per VIC-Register 29 Sprites in x-Richtung vergrößert, können sie nicht fließend hinter dem linken Bildschirmrand verschwinden, da sie zu breit sind (48 Pixel). Dort ist aber nur Platz für 24 Bildpunkte (die normale Sprite-Ausdehnung!). Dann verstecken sich die Sprites zwar bis zur Hälfte unter dem Randstreifen – doch werden sie dann plötzlich unsichtbar. Professionell sieht das gerade nicht aus.

Doch dazu gibt's einen Trick: Ist das Sprite halb verschwunden (also x-Position = 0), setzt man das dem Sprite entsprechende Bit in Register 16 und erhöht die x-Position auf 247. Verringert man diesen Wert jetzt Schritt für Schritt bis 224, löst sich das Sprite fließend auf.

Allerdings empfehlen wir, diese Routine in Assembler zu programmieren. Außerdem sollte man das Setzen des bewußten Bits in Register 16 per Raster-Interrupt außerhalb des Bildschirmfensters stattfinden lassen – sonst flimmert's unangenehm. (Oliver Kirwa/bl)

Paint Magic als Sprite-Editor

Wer das Malprogramm »Paint Magic« (64'er-Sonderheft 50) besitzt, kann es zum komfortablen Sprite-Editor umfunktionieren. Angenommen, man möchte überdimensionale Multicolorsprites erzeugen, die aus mehreren Einzel-Sprites bestehen, bei der Konstruktion aber nicht die Übersicht der Teil-Sprites verlieren.

Dazu brauchen Sie einen C64 mit Resetknopf und diese Software:
- Paint Magic (64'er-Sonderheft 50),
- SMON $C000 (64'er-Sonderheft 71) oder jeden anderen Maschinensprache-Monitor, der nicht den Bereich $6CC0 bis $6DC0 belegt.

Wenn Paint Magic aktiviert wurde, stellt man die gewünschten Grafikfarben ein und löscht den Bildschirm. Dann generiert man per BOX-Anweisung einen rechteckigen Bildschirmausschnitt (12 x 21 Pixel). Das Sprite-Muster läßt sich nun in diesem Window editieren. Möchten Sie mehrere Sprites aneinanderreihen, müssen Sie den Bildausschnitt (BOX) vergrößern. Nicht vergessen: fertige Grafik speichern!

Jetzt kommt der Trick: Per G-Befehl (Grab) schneidet man die Sprite-Muster aus dem Hires-Screen. Der Paint-Magic-Screen bietet allerdings nur Platz für maximal zwei Sprite-



Muster. Wenn das Sprite mit GRAB übernommen wurde, muß man den Resetknopf drücken. Laden Sie jetzt den Maschinensprache-Monitor und speichern Sie den definierten Sprite-Bereich von $6CC0 bis $6DC0.

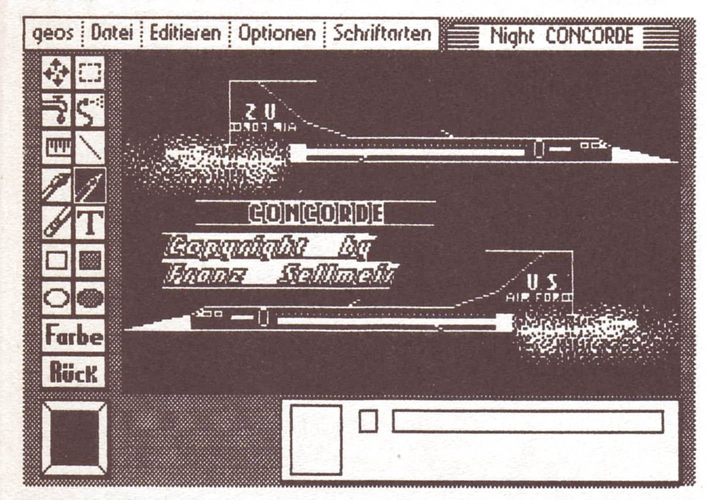
Besteht das vorgesehene Hyper-Sprite aus mehr als vier Einzelteilen, muß man die Prozedur entsprechend oft wiederholen. Zum Schluß liest man die Farbwerte aus dem Speicher, um die betreffenden Speicherstellen für eine spätere Weiterverarbeitung der Multicolor-Spritemuster richtig zu belegen:

- $5F40 (24384): Farbe der transparenten Pixel (Hintergrund),
- $5F41 (24385): Multicolorfarbe 0 ($D025),
- $5F42 (24386): Spritefarbe (Vordergrund),
- $5F43 (24387): Multicolorfarbe 1 ($D026).

(Aziz Ögüt/bl)

Grafik mit Geos

Geos-Grafiken in eigene Programm einbauen:
Hires-Bilder, die man z.B. mit Geopaint gezeichnet hat, sind nicht problemlos in eigenen Basic-Programmen oder den bekannten C-64-Zeichenprogrammen zu integrieren. Geos verwendet ein spezielles Grafik- und Speicherformat auf Diskette, das nichts mit dem Hi-Eddi-Standard gemeinsam hat. Wer aber ein Freeze-Modul besitzt (z.B. Final Cartridge, Action Replay usw.), ist fein raus: Per Tipp auf den entsprechenden Knopf läßt sich die aktuelle Bildschirmgrafik auf jeden Fall retten und anschließend auf Diskette sichern. Der Vorteil: Solche gescannten Hires-Grafiken kann man jederzeit mit konventionellen Zeichenprogrammen (z.B. Hi-Eddi, Starpainter usw.) laden und verarbeiten (Abb. 7).



[7] Diese Grafik wurde aus Geopaint geklaut und ins Hi-Eddi-Format verwandelt

Wer mit dem C-64-Modus des C 128 arbeitet, kann auf solche Steckmodule verzichten: Dazu reicht der Speicher des C 128 und der eingebaute Maschinensprache-Monitor Tedmon völlig aus. Wenn die Geopaint-Grafik auf dem Bildschirm steht, drücken Resetknopf, um in den 40-Zeichenmodus des C 128 zukommen: Die Pixel der Geos-Grafik liegen im RAM von $A000 bis $BFFF. Löschen Sie jetzt den normalen C-128-Hires-Speicher:

```
GRAPHIC 1,1: GRAPHIC 0
```

Aktivieren Sie nun den Tedmon (mit <F8> oder die Anweisung MONITOR) und verschieben Sie Grafik-Bytes in den Hires-Bereich:

```
T A000 BFFF 2000
```

Die Anweisung GRAPHIC 1 zeigt Ihnen, ob's mit der Übertragung geklappt hat. Sind Sie mit dem Ergebnis zufrieden, läßt sich der Hires-Speicher als normale Grafik im Hi-Eddi-Format auf Diskette sichern:

```
S "GEOS.PIC" 08 2000 3F40
```

oder in Basic 7.0:

```
BSAVE "GEOS.PIC",ON B0, P8192 TO P16191
```

Von Geopaint nach Geowrite:
Geowrite besitzt keine spezielle Menüfunktion, um Grafik ins Textdokument einzubinden. Das geht nur über einen Umweg:

Markieren Sie in Geopaint den gewünschten Grafikausschnitt mit dem Gummirechteck. Aktivieren Sie die Option »copy« im Edit-Menü, die Grafik wird mit dem Namen »Photo Scrap« auf Diskette gesichert.

Jetzt verläßt man Geopaint und startet Geowrite. Positionieren Sie den Cursor im Editorbildschirm an der Stelle, an der das Bild erscheinen soll. Wählen Sie den Punkt »paste« im Edit-Menü. Der Computer lädt nun »Photo Scrap« von der Disk und fügt die Grafik an gewünschter Stelle ein. »Photo Scraps« sollte man im Album »Fotomanager« sammeln, um sie bei späteren Gelegenheit jederzeit mit Geopaint oder Geowrite wieder zu verwenden.

Print-Shop- oder Hi-Eddi-Grafik mit Geos:
Zwei Möglichkeiten gibt's:

- Print-Shop-Grafiken lassen sich ohne weiteres mit dem Geos-Utility »Graphics Grabber« (auf der Deskpack-Disk) ins Geos-Format übertragen. Als Photo-Scrap kann man solche Bilder dann weiterverarbeiten.
- Hires-Screens von Hi-Eddi, Giga-CAD oder Printfox müssen vorher in eine Geos-Datei konvertiert werden. Das macht unser »Bitmap-Converter« im 64'er-Sonderheft 28. (bl)

Linierter Bildschirm

Unser Listing verwandelt den Textbildschirm in ein liniertes Notizblatt. Das ist nützlich, wenn man zeilengenau arbeiten will, z.B. bei der Eingabe eines Basic-Listings:

```
10 DIM HX(75)
20 FOR I=0 TO 9: HX(48+I)=I: HX(65+I)=I+10: NEXT
30 FOR I=49152 TO 49247: READ MC$
40 HI=ASC(LEFT$(MC$,1)): LO=ASC(RIGHT$(MC$,1))
50 DZ=16*HX(HI)+X(LO): POKE I,DZ: NEXT
60 SYS 49206: NEW
1000 DATA EA,EA,EA,EA,EA,A9,01,8D
1010 DATA 19,D0,A9,00,8D,21,D0,A2
1020 DATA 06,EA,EA,CA,D0,FB,A9,06
1030 DATA 8D,21,D0,A5,02,C0,F9,F0
1040 DATA 0B,18,69,08,85,02,8D,12
1050 DATA D0,4C,81,Ea,A9,39,85,02
1060 DATA 8D,12,D0,4C,31,EA,78,A9
1070 DATA 39,85,02,8D,12,D0,AD,11
1080 DATA D0,29,7F,8D,11,D0,A9,01
1090 DATA 8D,0D,DC,8D,1A,D0,A9,00
1100 DATA A2,C0,8D,14,03,8E,15,03
1110 DATA 58,60,4D,00,03,9D,80,7F
```

Die Bildschirmfarben lassen sich ändern:

- POKE 49175, 0 bis 15: Hintergrund,
- POKE 49163, 0 bis 15: Linienfarbe.

Kursiver Commodore-Zeichensatz

Es geht auch ohne Änderung des Original-C-64-Zeichensatzes: Durch geschickte Programmierung des VIC-Chip läßt sich der Effekt kursiver Schriftzeichen (Italic) realisieren:

```
10 DIM HX(75)
20 FOR I=0 TO 9: HX(48+I)=I: HX(65+I)=I+10: NEXT
30 FOR I=49152 TO 49239: READ MC$
40 HI=ASC(LEFT$(MC$,1)): LO=ASC(RIGHT$(MC$,1))
```

```
50 DZ=16*HX(HI)+X(LO): POKE I,DZ: NEXT
60 SYS 49197: NEW
1000 DATA EA,EA,EA,EA,EA,A9,01,8D
1010 DATA 19,D0,AD,16,D0,49,01,8D
1020 DATA 16,D0,A5,02,C9,F6,F0,0B
1030 DATA 18,69,04,85,02,8D,12,D0
1040 DATA 4C,81,EA,A9,32,85,02,8D
1050 DATA 12,D0,4C,31,EA,78,A9,32
1060 DATA 85,02,8D,12,D0,AD,11,D0
1070 DATA 29,7F,8D,11,D0,A9,01,8D
1080 DATA 0D,DC,8D,1A,D0,A9,00,A2
1090 DATA C0,8D,14,03,8E,15,03,58
1100 DATA 60,C0,8D,14,03,8e,15,03
```

Mit <RUN/STOP RESTORE> stellt man den Normalzustand wieder her.

Bunte Screen-Effekte

Leicht in eigene Programmentwicklungen einzubauen, erzeugt unser Listing auf dem Bildschirm bunte Streifen ,bis eine beliebige Taste gedrückt wird. Text, der bereits auf dem Bildschirm steht, wird dabei nicht verändert:

```
10 FOR I=49152 TO 49182: READ A: POKE I,A
20 =S+A: NEXT
30 IF S<>4479 THEN PRINT "FEHLER IN DATAS": END
40 SYS 49152
50 DATA 162,0,160,73,136,208,253,142,32,208,142,33,
208,232,208,242,165,198
60 DATA 240,236,162,14,160,6,142,32,208,140,33,
208,96
```

Mit SYS 49152 läßt sich der Bildschirm-Gag nach <RUN/STOP RESTORE> erneut aktivieren.

Newsroom-Grafiken klauen

Laden Sie dieses DTP-Programm, holen Sie die gewünschten Grafiken in den Speicher und drücken Sie den Resetknopf. Mit unserem Basic-Sechszeiler können Sie die Bitmap speichern, um Sie mit anderen Zeichenprogrammen nachzubearbeiten:

```
10 POKE 56,6*16: CLR
20 OPEN 1,8,2,"(Dateiname),P,W"
30 PRINT#1,CHR$(0);CHR$(32);
40 FOR I=6*4096 TO 8*4096
50 PRINT#1,CHR$(PEEK(I));
60 NEXT I: CLOSE 1
```

Damit sichert man 8 KByte im Bereich von $6000 (24576) bis $7FFF (32767).

Kaleidoskop im Lores-Bildschirm

Per Zufallsgenerator bringt unser kurzes Basic-Listing Kaleidoskop-Muster auf den Screen. Per Leertaste kann man den Bildschirm löschen, bevor er sich mit einem anderen Muster erneut aufbaut:

```
5 PRINTCHR$(147)CHR$(142): Z=160
6 POKE 53280,0: POKE 53281,0
10 C=INT(RND(0)*15)
20 X=INT(RND(0)*40)
30 Y=INT(RND(0)*25)
40 :
50 E1=1024: E2=1063: E3=1984: E4=2023
60 F1=55296: F2=55335: F3=56256: F4=56295
70 :
80 POKE E1+X+40*Y,Z
85 POKE F1+X+40*Y,C
90 POKE E2-X+40*Y,Z
95 POKE F2-X+40*Y,C
100 POKE E3+X-40*Y,Z
105 POKE F3+X-40*Y,C
110 POKE E4-X-40*Y,Z
115 POKE F4-X-40*Y,C
130 IF PEEK(203)=60 THEN PRINTCHR$(147);
140 GOTO 10
```

Es muß nicht immer Hires-Grafik sein, um tolle Farbeffekte auf den Screen zu zaubern! (bl)

Kürzer geht's nicht...

Hier ist ein Sprite-Editor für den C64, der aus nur zwei Basic-Zeilen besteht:

```
10 FOR I=0 TO 2: A=0: FOR N=0 TO 7:
A=A-2↑(7-N)*(PEEK(1024+40*X+N+8*I)=42):
NEXT: A(I+1)=A
20 NEXT: PRINT TAB(25)A(1)A(2)A(3): X=X+1:
IF X<21 THEN 10
```

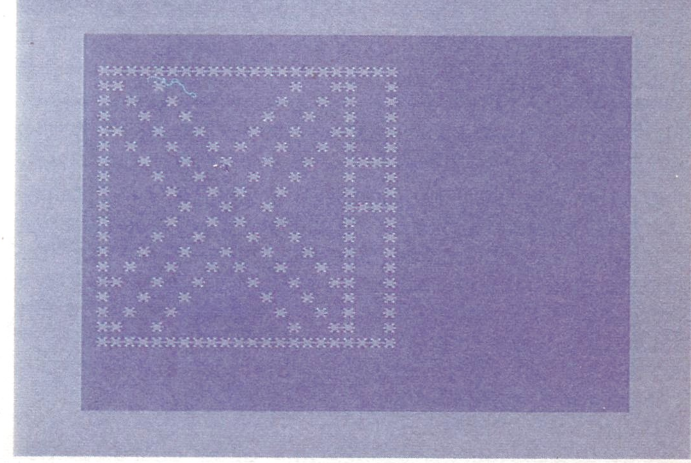
Speichern Sie das Mini-Listing auf Disk, starten Sie aber noch nicht mit RUN! Bleiben Sie im Direktmodus des C64 und löschen Sie den Bildschirm (per <SHIFT/CLR HOME>). Fangen Sie nun in der linken oberen Ecke mit dem Entwurf des Sprite-Musters an: (wir empfehlen, die oberste Zeile nicht zur Sprite-Definition zu verwenden und stattdessen diese übersichtliche Zahlenreihe einzutippen):

```
123456789012345678901234
```

Damit wissen Sie auf Anhieb, welches Pixel man an der entsprechenden Stelle im Sprite setzen muß. Die Zahlenreihe wird beim Lesen der Sprite-Werte übergangen, da kein CHR$(42) darin vorkommt!

- <*> setzt ein Pixel, <SPACE> löscht oder überspringt die Position des Bildpunkts. Denken Sie daran, daß man horizontal 24 Spalten und vertikal 21 Zeilen verwenden muß, um eine korrektes Sprite-Raster zu erzeugen (Abb. 8). Mit den Cursor-Tasten kann man sich im gesamten Editorfeld bewegen und das Sprite beliebig ändern.

Sind Sie mit dem Entwurf zufrieden, müssen Sie den Cursor in der vorletzten Zeile des Textbildschirms positionieren, RUN eingeben und <RETURN> drücken. Damit startet



[8] Der kürzeste Sprite-Editor der Welt verwandelt den Bildschirm zum Zeichenblatt

man den Basic-Zweizeiler: Er beginnt mit der Rechenarbeit. Am rechten Bildschirmrand erscheinen die entsprechenden DATA-Werte, die man notieren oder als DATA-Zeilen in eigene Basic-Programme übernehmen sollte.

(Daniel Wicker/Dr. L. Meyding/bl)



Fortsetzung von Seite 6

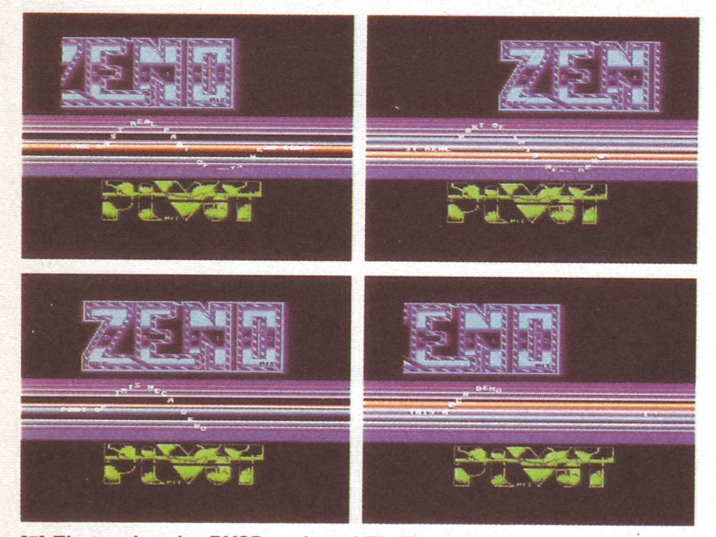
schirm geschriebenen Screen-Zeilen ist unsere »FLASH«-Routine zuständig:

```
FLASH      LDY #$77         ;3LINES=$77 CHARS
           LDX FLASHCOUNT;COUNTER HOLEN
           LDA FLASHTAB,X;UND BYTE
FLASH1     STA $DA08,Y      ;SCHREIBEN SOLANG
           DEY              ;BIS ALLE DREI
           BPL FLASH1       ;ZEILEN FERTIG
           INC FLASHCOUNT;COUNTER+1
           CPX #$18         ;SCHON 18 WERTE?
           BNE FLASHEND     ;NEIN DANN ENDE
           LDA #$00         ;JA DANN ZAEHLER
           STA FLASHCOUNT;ZURUECKSETZEN
FLASHEND   RTS              ;ENDE

FLASHTAB  ;FARBWERTE
          .BYTE $02,$02,$02,$0A,$0A,$0A
          .BYTE $07,$07,$07,$0F,$0F,$0F
          .BYTE $07,$07,$07,$0A,$0A,$0A
          .BYTE $02,$02,$02,$00,$00,$00
```

Diese schreibt die jeweils aktuelle Farbe 120 Spalten lang ins Farb-RAM und kehrt danach in die Hauptschleife zurück. Da beim nächsten Aufruf dieser Routine auch der nächste Farbwert an die Reihe kommen muß, brauchen wir eine Speicherstelle, die wir inkrementieren (»FLASHCOUNT«). Auch hier greift wieder das Single-Step-Prinzip: **eine** Farbe schreiben und danach zurück, beim nächsten Aufruf die nächste usw. Die Farbwerte sind übrigens aus Verzögerungsgründen dreimal vorhanden. Stünde jeweils nur ein Byte, wäre das Flashing viel zu schnell.

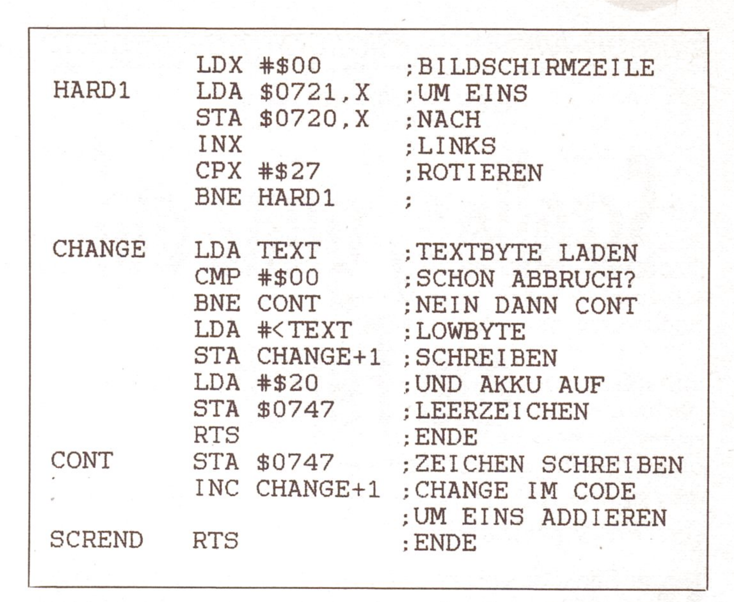
Ab Label »ST3« in unserer IRQ-Schleife warten wir auf Rasterzeile $B8 und das aus gutem Grund: ab dieser Zeile soll unser Screenscrolling losgehen (»CHARSCROLL«).



[5] Ein tauchender DYCP und zwei Tic Tacs sorgen für Stimmung

```
CHARSCROLL
          LDX SCRHELP     ;REGISTER HOLEN
          DEX             ;UND ZWEIMAL
          DEX             ;DEKREMENTIEREN
                          ;(SCROLLSPEED)
          STX SCRHELP     ;IN HILFS & SCRREG
          STX SCROLLREG;SCHREIBEN
          CPX #$BF        ;SCHON UNTERLAUF?
          BEQ HARDSCR     ;DANN HARDSCROLL
          RTS             ;ENDE

HARDSCR   LDX #$C7        ;REGISTER
          STX SCROLLREG;ZURUECK-
          STX SCRHELP     ;SETZEN
```



```
          LDX #$00        ;BILDSCHIRMZEILE
HARD1     LDA $0721,X     ;UM EINS
          STA $0720,X     ;NACH
          INX             ;LINKS
          CPX #$27        ;ROTIEREN
          BNE HARD1       ;

CHANGE    LDA TEXT        ;TEXTBYTE LADEN
          CMP #$00        ;SCHON ABBRUCH?
          BNE CONT        ;NEIN DANN CONT
          LDA #<TEXT      ;LOWBYTE
          STA CHANGE+1    ;SCHREIBEN
          LDA #$20        ;UND AKKU AUF
          STA $0747       ;LEERZEICHEN
          RTS             ;ENDE
CONT      STA $0747       ;ZEICHEN SCHREIBEN
          INC CHANGE+1    ;CHANGE IM CODE
                          ;UM EINS ADDIEREN
SCREND    RTS             ;ENDE
```

Dazu müssen wir zunächst unser Hilfs-Byte entsprechend dekrementieren und anschließend ins Hardscroll-Register schreiben. Ist das genau achtmal geschehen (acht Bits nach links geschoben), setzen wir die Register wieder auf den ursprünglichen Wert und ziehen die komplette Bildschirmzeile um ein Byte nach links. Jetzt schieben wir nur noch ein neues Zeichen von rechts in den Bildschirm und fertig. Damit dieser Scroller nicht zu eintönig aussieht, lassen wir wieder ein paar Farben rotieren (»CHARFLASH«).

Hier benutzen wir dieselbe Technik wie beim Raster-Bar-Scrolling: einfach den Farb-RAM-Bereich um eine Stelle nach links rotieren und den ersten Farbwert auf die letzte Spalte setzen.

```
;-----------------------------------------

CHARFLASH
          LDY $DB20       ;FARBRAM VON
          LDX #$00        ;$DB20 BIS
CHARFL1   LDA $DB21,X;$DB47
          STA $DB20,X;ENDLOS
          INX             ;ROTIEREN
          CPX #$27        ;DAS Y-REG DIENT
          BNE CHARFL1:ALS ZWISCHEN-
          STY $DB47       ;SPEICHER
          RTS             ;ENDE

FLASHTAB2 ;FARBTABELLE
          .BYTE $06,$06,$06,$04,$04,$04
          .BYTE $0E,$0E,$0E,$03,$03,$03
          .BYTE $0F,$0F,$0F,$01,$01,$01
          .BYTE $01,$01,$01,$0F,$0F,$0F
          .BYTE $03,$03,$03,$0E,$0E,$0E
          .BYTE $04,$04,$04,$06,$06,$06
          .BYTE $00,$00,$00,$00
TABEND2
```

Sollte bei Ihren Demo-Bemühungen der Bildschirm des öfteren flackern, braucht irgendeine Routine mehr Zeit, als Rasterzeit zur Verfügung steht. Beschränken Sie z.B. den Bereich eines Scrollings auf weniger als vier Rasterzeilen, kommt's sicher zum Crash. Teilen Sie Ihren Routinen anfangs also lieber etwas mehr Zeit zu (ausprobieren) und minimieren Sie diese später durch Austesten der benötigten Zeit, indem Sie einfach den Rasterbereich Schritt für Schritt einschränken.

Übrigens: erwarten Sie nicht, auf Anhieb professionelle Demos wie in den Abbildungen 2, 3, 4 und 5 zusammenzuschustern; es ist bis heute noch kein Meister vom Himmel gefallen...

(pk)

Screen-Mover – Bildschirmbewegung der sanften Art

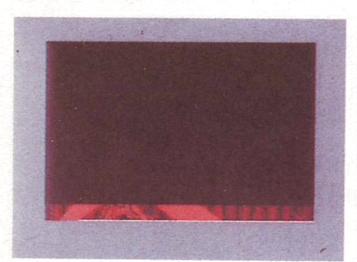
Scrollen, ganz auf die Feine

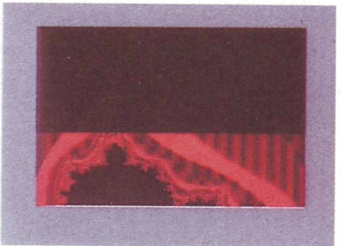
Serienmäßig ist Scrollen beim C 64 nur ruckweise und nach unten kaum möglich. Das wird mit diesem Programm anders.

Der Effekt des pixelweise nach oben und unten scrollenden Bildschirms ist aus kommerziellen Spielen bekannt, wo mit dieser Methode Highscore-Listen oder höhere Level auf den Bildschirm gebracht werden.

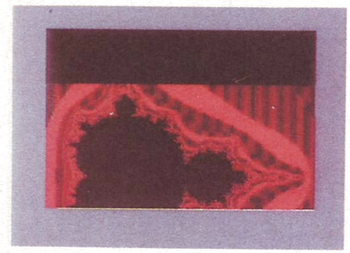
Daß dieses Verfahren gar nicht so aufwendig ist, beweist unser Programm »Screen Mover« mit seinen nur 184 Byte. Sie können es daher leicht in eigene Programme einbauen.

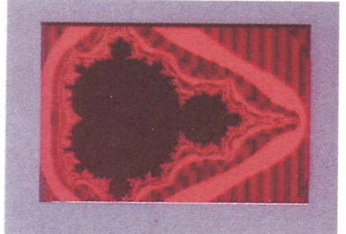
Durch bestimmte Raster-Interrupt-Techniken wird, ohne den Grafikspeicher manipulieren zu müssen oder andere aufwendige Berechnungen, der Effekt erreicht. Da die Routine in den Interrupt eingebaut ist, kann man sich ihrer problemlos in eigenen Programmen bedienen, beispielsweise, um eine Diashow so richtig professionell zu gestalten.





Das Demoprogramm läßt eine Hires-Grafik auf- und abscrollen (Beispiel: Apfelmännchen)





Das Programm können Sie mit

```
LOAD "SCR.MOVER $C000",8,1
```

laden und nach Eingabe von NEW starten. Geben Sie dazu

```
SYS 49152
```

ein. Mit einem an Port 2 angeschlossenen Joystick können Sie jetzt den Bildschirm nach Belieben scrollen. Möchten Sie nicht den Text- sondern den Grafikbildschirm scrollen, aktivieren Sie diese mit

```
POKE 49333,48
POKE 53272,29
```

Das Programm erwartet die Grafik ab Adresse $2000 (HI-Eddi-Format). Die Verschiebung funktioniert genauso wie im Textmodus.

Sprites können mit diesem Verfahren leider nicht mitgescrollt werden. Sind Sprites eingeschaltet, flackert aufgrund des geänderten Timings der Bildschirm.

Wenn Sie die Routine in einem eigenen Programm aufrufen möchten, können Sie mit

```
POKE 49334,X
```

die Bildschirmhöhe eingeben. Bildschirmhöhe bedeutet hier die sichtbare Höhe, die auf dem Bildschirm angezeigt wird. X kann zwischen 3 und 194 liegen.

Die Hintergrundfarbe, also die Farbe der Fläche über dem eigentlichen Text- oder Grafikbild, bestimmen Sie mit

```
POKE 49332,F
```

wobei F Werte zwischen 0 und 15 annehmen darf.

Noch ein paar Anmerkungen zu den Besonderheiten des VIC: Der Video-Controller des C64 benötigt für Rastertechniken dieser Art eine Angabe über das Füllmuster, das er im nicht benutzten Bereich, also im oberen Teil des Bildschirms, verwenden soll. Merkwürdigerweise verlangt er diese Angabe in Speicherstelle $3FFF (dez. 16383), die aber mitten im Basic-Speicher liegt. Screen Mover setzt in diese Adresse zu Beginn eine 0, damit die ungenutzte Fläche auch leer erscheint. Bei längeren Basic-Programmen, die über $3FFF hinausreichen, kann daher ein Byte des Programms oder des Variablenspeichers überschrieben werden. Die Folge ist ein Absturz oder Fehlfunktionen. Achten Sie also darauf, daß dieser Bereich von Ihrem Programm nicht beeinflußt wird. Sie können beispielsweise das Ende des Basic-Speicher mit diesem Befehl herabsetzen:

POKE	Funktion
POKE 49333,16: POKE 53272,21	Textmodus einschalten
POKE 49333,48: POKE 53272,21	Hires-Grafik ein
POKE 49332,0 bis 15	Hintergrundfarbe
POKE 49334,3 bis 194	Bildschirm verschieben

Tabelle 1. Wichtige Befehle zur Steuerung des Screen Movers

Adresse	Inhalt
$C000 bis $C020	Rasterinterrupt initialisieren
$C021 bis $C02F	wartet die erste Rasterzeile ab ($29)
$C031	Einsprung in die Blitter-Routine
$C034 bis $C050	hat der Rasterstrahl den unteren Bildschirmrand erreicht, wird der Rahmen abgeschaltet
$C053 bis $C06D	Routine zum Blittern des Screens. Das Bild wird bei jeder Rasterzeile stückweise nach unten verschoben ($D011). Das wird durch die Häufigkeit der Durchläufe bestimmt.
$C06E bis $C095	Abfrage des Joysticks Port 2
$C098 bis $C0B3	Farben setzen
$C0B4 bis $C0B8	Scroll- und Farbspeicher

Tabelle 2. Die Speicherbelegung des Programms

```
POKE 56,63:CLR
```

Damit haben Sie dann noch ca. 13 KByte zur Verfügung.

Weitere interessante Effekte können Sie mit

```
POKE 16383,X
```

erzielen (X zwischen 0 und 255). So franst beispielsweise

```
POKE 16383,170
```

den oberen Bildschirmrand aus.

Weitere Informationen über verwendete Speicherbereiche, die Adressen der einzelnen Routinen oder interessante Speicherstellen finden Sie in den beiden Tabellen.

Wir hoffen, daß Sie mit dieser Super-Routine interessante Effekte produzieren und viele Zweifler verblüffen werden, probieren Sie's aus.

(hb)

Kurzinfo: Screen Mover

Programmart: Grafiktool
Laden: LOAD "SCR.MOVER",8,1
Starten: SYS 49152
Besonderheiten: kein Sprite-Scroll möglich
Benötigte Blocks: 1
Programmautor: Joachim Hirth



Grafikkonverter

Printfox in IFF

Printfox-Grafiken gibt es wie Sand am Meer. Bislang konnte man diese Bilder aber auf dem Amiga oder PC nicht weiterverwenden. Der Printfox-IFF-Konverter schließt diese Lücke und schafft eine Brücke zwischen den Systemen.

Mit dem Tool werden Bilder und Grafiken im Printfox-Format ins IFF-Format transformiert. IFF oder auch LBM, ist das Standardformat auf dem Amiga. Es kann auch von einem PC verarbeitet werden. Die Umrechnung der Grafik erfolgt im C64. Das IFF-File wird im C-64-Format auf Diskette geschrieben und muß auf den Amiga oder PC übertragen werden. Dazu gibt's mehrere Wege:
- Nullmodemkabel und Terminalprogramm (Amiga und PC)
- mit den Tools Janus oder BDOS (PC) - 64'er 2/90
- mit dem Grafikkonverter »Big-VIC« (Amiga) - 64'er 2/92
- RAMTrans (Amiga) - 64'er 10/92

Ist das Programm geladen und gestartet, fragt der Computer nach dem Namen des zu konvertierenden Programms (s. Abb 1), der Laufwerksnummer, von wo geladen werden soll, nach dem Namen des umgerechneten IFF-Files und der Laufwerksnummer für das Speichern. Dann wird das zu bearbeitende Bild geladen und umgerechnet.

Hat man das File auf das andere Computersystem übertragen (Abb. 2 und 3), können die Bilder ohne Probleme weiterverarbeitet werden. Die Eingaben für Namen und Laufwerke sind mit dem Standard-INPUT des C64 programmiert und ohne Sicherheitsabfrage. Da dieser Teil des Tools in Basic geschrieben wurde, bleibt genügend Raum für eigene Gestaltung. (lb)

```
      PRINTFOX-IFF CONVERTER
    RAINER THEUERKORN JUNI 91

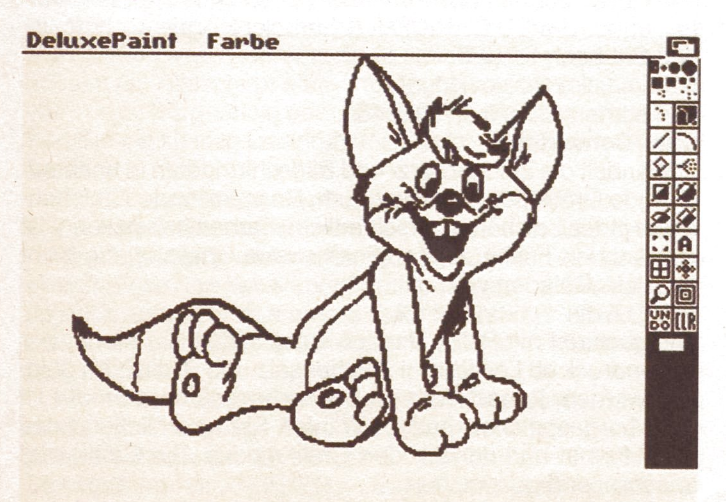
 QUELLFILENAME:? FUCHS1.GB
 QUELLGERAETEADRESSE:? 8

 ZIELFILENAME:? FUCHS.IFF
 ZIELGERAETEADRESSE:? 8

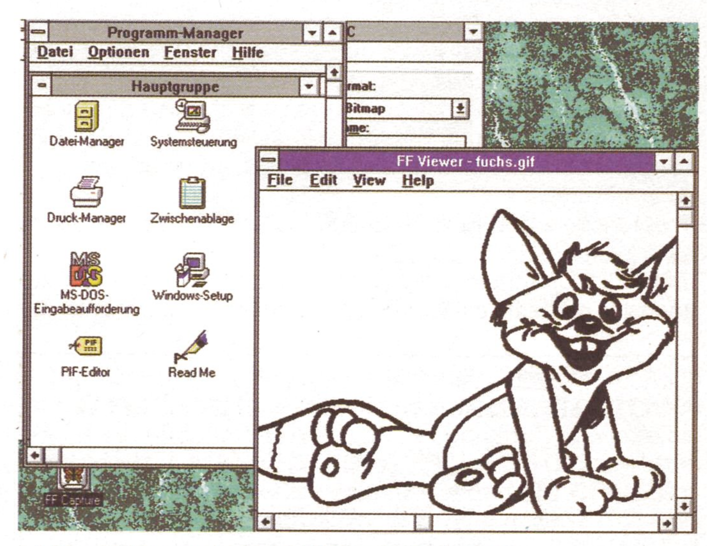
 FILE WIRD GELADEN
 STATUS: 0 OK 0  0

 DISKETTE EINLEGEN UND TASTE DRUECKEN!
```

[1] Das Eingabemenü des Konverters



[2] Der Printfox in DPaint auf dem Amiga



[3] Der schlaue Fuchs unter Windows auf dem PC

Kurzinfo: Printfox-IFF-Konverter

Programmart: Grafikkonverter
Laden: LOAD"IFF-Konverter",8,1 oder LOAD"LOADERCON",8,1
Starten: nach dem Laden <RUN> eingeben
Besonderheiten: keine Sicherheitsabfragen bei Eingabe
Benötigte Blocks: 14
Programmautor: Rainer Theuerkorn

Dateien-Export

Das Übertragen des konvertierten Printfox-Files auf den Amiga ist mit dem Tool »Big Vic« nur unter der Betriebssystemversion 1.3 möglich, da unter neuen Systemen (OS 2.0/3.0) die Software nicht läuft und der Amiga ins Nirvana geschickt wird. Eine angepaßte Version existiert momentan nicht.

Das Programm »RAMTrans« arbeitet sowohl mit alten Amiga-Betriebssystemversionen zusammen, als auch mit den Neuen. Besitzer eines Amiga 1200 können dieses Tool ebenfalls nutzen, sollten aber den Cache beim Booten abschalten. Eine Version von »RAMTrans« für MS-DOS ist in Vorbereitung. Bei beiden Programmen ist der Einsatz von Turbo-Boards mit Vorsicht zu genießen, denn die schnelleren Prozessoren können zum Absturz führen.

Die IFF-Dateien können, nach dem Übertragen auf den Amiga, auf MS-DOS-Disketten geschrieben werden (Cross-DOS o.ä.) und einer Weiterverarbeitung auf Archimedes, Mac, Atari ST oder PC steht nichts im Wege.

Picture Tool 1.0 – Grafik oder Zeichensatz suchen und sichern

Knopfdruck genügt

Ob Multicolor, Hires oder Zeichensatz, die Programme von »Picture Tool« finden alles.

Grafiken von Profispielen werden mit größeren Computern designt. Was dabei entsteht ist wirklich faszinierend. Doch was nützt das tollste Scenario, wenn Sie es nur für einen kurzen Augenblick im Spiel betrachten können. Picture Tool 1.0 macht Schluß mit dieser Schonkost. Wenn die Grafik erscheint, drücken Sie RESET.

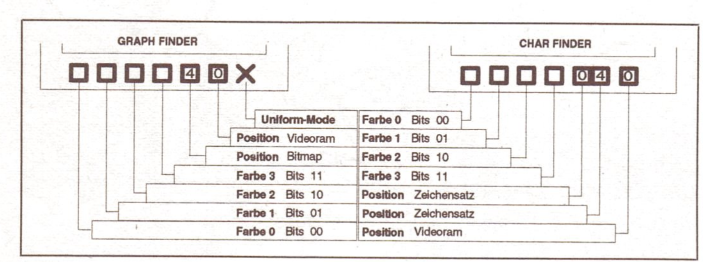
Danach laden Sie für Grafiken eines der beiden »GRAPH FINDER«- oder für Bildschirme eines der »CHAR FINDER«-Tools, passen die Farben an und sichern die Daten auf Disk. Später lassen sich diese mit »GRAPH FORMATTER« (Grafiken) an die Formate Koalapainter, Paint Magic, Art Studio, Hires oder Multicolor anpassen. Mit »CHAR-CONVERTER« wandeln Sie Bildschirm und Zeichensatz in Hires um.

Graph-Finder-Lo

... durchsucht den Bereich $4000 bis $FFFF nach Grafiken.

```
LOAD"GRAPH FINDER LO",8
```

und mit RUN starten. Im Arbeitsbildschirm sehen Sie zunächst in der Mitte ein willkürliches Pixel-Muster. Darunter die



Die Informationsfelder von »Graph Finder«

Kurzinfo: Graph Finder

Programmart: Grafik-Grabber
Laden: LOAD"GRAPH FINDER LO",8
Starten: nach dem Laden RUN eingeben
Laden: LOAD"GRAPH FINDER HI",8,1
Starten: nach dem Laden SYS18546 eingeben
Benötigte Blocks: 10/9
Programmautor: Norbert Nagy

Informationsfenster (Abb.). Sie geben Aufschluß über Farben, momentane Position und einem Uniformmodus (wenn aktiv, erscheint rechts unten »X«). In dieser Funktion werden die Farben der linken oberen Ecke für das ganze Bild übernommen.

<**F1**> ... zählt den Darstellungsbereich in 8-KByte-Schritten.
<**F3**> ... wählt das Video-RAM in 1-KByte-Schritten.
<**F5**> ... wechselt zwischen Hires und Multicolor.
<**F7**> ... Uniform wird eingeschaltet. »X« rechts zeigt diese Funktion an. Wenn Sie mit <**F1**> die Bank wechseln, wird neu ausgelesen und »Uniform« muß neu aktiviert werden.
<**F2**> ... ändert die Hintergrundfarbe (Bits 00).
<**F4**> ... wechselt Farbe 1 (Bits 01).
<**F6**> ... ändert Farbe 2 (Bits 10).
<**F8**> ... wechselt Farbe 3 (Bits 11).
<**RETURN**>
... speichert das Bild im »GFL«-Format (Grafik Lo) unter dem Namen »SAVED PIC.GFL«.

Graph-Finder-Hi

... durchsucht den Bereich $0000 bis $3FFF und $8000 bis $FFFF nach Grafiken. Die Files werden als »SAVED PIC.GFH« (Grafik Hi) gespeichert. Laden Sie nach RESET

```
LOAD"GRAPH FINDER HI",8,1
```

und starten Sie mit SYS18546.

Graph Formatter

Geladen wird mit:

```
LOAD"GRAPH FORMATTER",8
```

und gestartet mit RUN. Danach entscheiden Sie per Tastendruck, ob eine Lo- oder Hi-Grafik geladen wird. Wählen Sie anschließend das Format aus und geben Sie den File-Namen ein. Die Grafik wird gespeichert.

Zum Suchen nach Bildschirminhalten und Zeichensätzen stehen zwei weitere Tools zur Verfügung.

Char-Finder-Lo

... durchsucht den Bereichs $4000 bis $FFFF. Laden Sie dazu nach RESET mit

```
LOAD"CHAR FINDER LO",8
```

und starten Sie mit RUN. Der Arbeitsbildschirm ist ähnlich dem des Grafik-Finders aufgebaut (Abb.). Auch hier sehen Sie zuerst ein willkürliches Bildschirmmuster. Darunter die Informationsfenster. Sie zeigen Farben und Position von Zeichensatzgenerator und Video-RAM:

<**F1**> ... wählt Zeichensatz in 2-KByte-Schritten aufwärts.
<**F3**> ... ändert das Video-RAM in 1-KByte-Schritten (innerhalb der aktuellen 16-KByte-Bank).
<**F5**> ... wechselt zwischen Normal, Multicolor und Extended (die Anzeigen für die Farben werden angepaßt).
<**F7**> ... ändert die Zeichenfarbe
<**F2**> ... wechselt die Hintergrundfarbe (Bits 00).
<**F4**> ... ändert Farbe 1 (Bits 01).
<**F6**> ... wechselt Farbe 2 (Bits 10).
<**F8**> ... ändert Farbe 3 (Bits 11).
<**RETURN**>
... speichert Zeichensatz, mit Bildschirminhalt im »CFL«-Format (Character Lo) unter den Namen »SAVED VID.CFL« und »SAVED CHR.CFL«. Sollten schon Files unter diesem Namen auf Diskette sein, werden diese überschrieben.
<**SHIFT RETURN**>
... speichert nur den Zeichensatz.

Char-Finder-Hi

... entspricht in der Bedienung dem Finder-Lo und durchsucht den Bereich $0000 bis $3FFF und $8000 bis $FFFF nach Bildschirminhalten und Zeichensätzen. Die Files werden unter »SAVED CHR.CFH« (Character Hi) gespeichert. Laden Sie nach RESET

```
LOAD"CHAR FINDER HI",8,1
```

und starten Sie mit SYS18546.

Char Converter

Wandelt die Zeichensatz- und Bildschirmdaten in hochauflösende Grafik oder Multicolor um. Da im extended Zeichenmode jedes Zeichen unterschiedliche Farben haben kann, ist in manchen Fällen eine originalgetreue Umwandlung nicht möglich. Geladen wird mit:

```
LOAD"GRAPH FORMATTER",8
```

und gestartet mit RUN. Danach entscheiden Sie wieder per Tastendruck ob Lo- oder Hi-Zeichensatz und Bildschirm geladen werden, je nachdem ob Sie vorher mit dem Finder Hi oder Lo gespeichert haben. Wählen Sie anschließend das Grafikformat und danach den Grafikmodus. Die Grafik wird anschließend gespeichert.

Zum Schluß noch ein Tip: Wenn Sie

```
LOAD"PICTURE TOOL 1.0",8
```

laden und mit RUN starten, finden Sie Informationen über den Programmautor. (Herbert Großer)

GLD V2.0 – Supertool für Zeichensatzgrafiken

Masken, Spiele, Landschaften

Übergroße Scenarien mit einer Breite von 256 Zeichen und 25 Zeilen Höhe lassen sich mit »Game Level Designer« komfortabel gestalten – ein Zeichensatz-Editor ist eingebaut.

Autos fahren durch Städte, Flugzeuge fliegen über weite Landschaften. Bis jetzt war es schwierig, solche Scenarien zu entwickeln. Mit Game Level Designer erhalten Sie ein komplettes Konstruktions-Set. Es erlaubt nicht nur das Neugestalten oder Bearbeiten eines Multicolor-Zeichensatzes, sondern zusätzlich auch einer kompletten Spielelandschaft mit 256 Zeichen Länge und 25 Zeilen Höhe.

Geladen wird von der beiliegenden Diskette mit:

```
LOAD"GLD V2.0/BY AMOK",8
```

und gestartet mit RUN. Danach sehen Sie das Hauptmenü. Um einen Eindruck von der Leistungsfähigkeit zu bekommen, laden Sie mit <**3**> zuerst »CHARSET«, dann mit <**4**> »OBJEKT SCREEN« und mit <**5**> »LEVEL«. In allen Lademenüs werden Sie zuerst nach dem Namen gefragt. Dieser muß buchstabengetreu eingegeben werden! Nach Return beginnt der Ladevorgang.

Wenn Sie danach mit <**1**> den Level-Editor aufrufen, sehen Sie eine Städtelandschaft (Abb.).

Level-Editor

Der Level-Editor ist in Objekt Screen und Level Screen gegliedert. Nach <**1**> befinden Sie sich im Objekt Screen.

Objekt Screen

Dieser Bildschirm erlaubt das Auswählen von Grafikelementen, die später im Level Screen zu einer Landschaft zusammengesetzt werden. Da ja schon einer geladen ist, erscheint am Bildschirm eine Anordnung von Einzelelementen wie Türen, Bürgersteig oder einem Tank. Ganz links oben sehen Sie ein kleines Quadrat, der Cursor. Er läßt sich mit dem Joystick in Port 2 bewegen.

Zurück zum **Hauptmenü** geht's mit <RETURN>.

Zur **Farbänderung** dienen die Tasten <1> bis <4>.

Die **Objektauswahl** treffen Sie, indem Sie auf der linken oberen Ecke den Feuerknopf drücken und anschließend per Joystick auf die rechte untere Ecke fahren (der erste Cursor bleibt stehen). Anschließend geht's zum Level Screen.

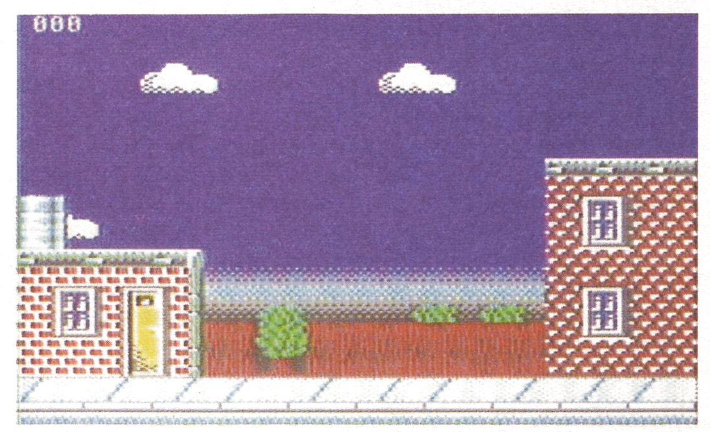
Level Screen

Hier sehen Sie das gewählte Objekt links oben, zusätzlich blinkt eine dreistellige Zahl. Sie gibt (dezimal) die Spaltenposition an. **Plazieren** läßt sich ein Objekt über den Joystick und **Einkopieren** mit <**FEUER**>. Eine **Farbänderung des Objekts** ermöglicht <**F1**>. Allerdings wird nur die Zeichenfarbe geändert. Zum **Bewegen des Level-Fensters** dienen <**F5**> (nach links) und <**F7**> (nach rechts). Eine direkte Anwahl der Level-Fenster erreichen Sie über <**1**> bis <**6**>. Löschen des gesamten Levels erreichen Sie mit <←>, wogegen Sie mit <**CLR**> nur **den Bildschirm löschen**. Zwei Zusatzfunktionen stehen zur Verfügung:

<**C**> **färbt den Level** und <**F**> **übernimmt linkes oberes Zeichen für den gesamten Level**. Zusätzlich läßt sich ein **Snapshot** des Bildschirms erreichen. Dazu drücken Sie <*>. Dieser Snapshot läßt sich mit <↑> an jede beliebige Stelle des Levels duplizieren und mit <**S**> betrachten. Er bleibt auch nach **Verlassen des Levelscreens** mit <**RETURN**> erhalten.

Aus dem Hauptmenü läßt sich der Snapshot auf Diskette sichern und als Object Screen laden. Dadurch kann man aus seinen Elementen neue Level konstruieren.

Er erlaubt die Konstruktion eines neuen Zeichensatzes. Ausgewählt wird er aus dem Hauptmenü mit <**2**>.



Fantastische Level lassen sich mit »Game Level Designer« konstruieren

Der Bildschirm ist in zwei Bereiche eingeteilt: In der Mitte ist das eigentliche Editierfeld, in dem in einem gepunkteten Raster die Änderungen durchgeführt werden. Pixelweise bewegt wird per Joystick, zusätzlich bringen die Cursor-Tasten jeweils ein Zeichen weiter. **Punkte** werden mit <**FEUER**> gesetzt. Die **Zeichenfarbe** läßt sich über <**SHIFT 1**> bis <**SHIFT 4**> ändern und wird mit <**1**> bis <**4**> als Zeichenfarbe übernommen. **Löschen** läßt sich ein Zeichen mit <**CLR/HOME**> und der gesamte Zeichensatz mit <**SHIFT CLR/HOME**>. Einzelne Pixel löschen Sie durch Setzen von

Charset Editor

Punkten in der Hintergrundfarbe. Im Zeichensatz lassen sich **Blocks definieren**: Dazu fahren Sie per Joystick auf die linke obere Ecke des gewünschten Bereichs und drücken <**B**>. Danach fahren Sie auf die rechte untere Ecke und drücken <FEUER>. **Kopieren** läßt sich dieser Block, indem Sie an die gewünschte Position fahren und <**C**> drücken. Die Zeichen eines definierten Blocks können Sie **spiegeln** wenn Sie <**X**> oder <**Y**> drücken. Mit <**G**> aktivieren Sie ein **Hilfsraster**. Ein **Zwischenspeicher** übernimmt mit <*> ein Zeichen und kopiert es mit <↑> an eine neue Cursor-Position. Wenn Sie den Zeichensatz das erste Mal verwenden, sollten Sie ihn mit der **Transfer**-Funktion (<**O**>) als Objekt Screen installieren. **Verlassen** wird der Charset-Editor mit <**RETURN**>. (Zeichensatz vorher speichern!)

Normalerweise ist die Konstruktion eines neuen Zeichensatzes im Charset-Editor der erste Arbeitsschritt. Installieren Sie ihn anschließend als Object Screen. Im Level-Editor stellen Sie sich jetzt aus den Einzelzeichen einen oder mehrere Level-Screen-Bildschirm(e) zusammen und definieren der Reihe nach einen als Snapshot, speichern diesen usw. Später läßt sich jeder Snapshot als Objekt Screen laden und seine Objekte in die Level einbauen. (Herbert Großer)

Sprite-Eddi 864 – konstruieren und animieren

Klein, bunt und bewegt

Das Handbuch des C 64 empfiehlt zur Gestaltung von Sprites ein kariertes Blatt Papier und viel Geduld. Wir dagegen empfehlen »Sprite-Eddi 864«. Mit ihm geht alles wie von selbst.

Wenn Sie bei Erwähnung von Bits und Bytes schon knapp vor einer Neurose sind, ist dieses Programm genau der richtige Tranquilizer. Lehnen Sie sich in Ihrem Computersessel zurück und lassen Sie Ihre Kreativität über sich kommen.

»Sprite Eddi 864« hat nicht nur alle wichtigen Befehle zum Bearbeiten und Verändern im Programm, sondern erlaubt dies sogar bei 864 Sprites im Speicher – und damit sie nicht zu statisch wirken, erweckt eine Animationsroutine die kleinen Bilder zum Leben.

Zuvor müssen Sie mit

```
LOAD"SPRITE-EDDI 864",8
```

den Editor laden und danach mit RUN starten. Der jetzt sichtbare Informationstext läßt sich mit <SPACE> abbrechen. Gleich darauf sehen Sie den Arbeitsbildschirm (Abb.). In der oberen Hälfte erscheinen acht Sprites, die zunächst alle das gleiche Aussehen haben, da sie auf denselben Arbeitsbereich zeigen.

In der unteren Hälfte sehen Sie zwei vergrößerte Sprites. Das linke dient als Vorlage für Verknüpfungen. Das rechte läßt sich bearbeiten. Rechts daneben wird ein Teil der Befehlsauswahl sichtbar. Die zuletzt gewählte Option ist mit einem weißen Balken versehen. Darüber zeigt »BL:000«, daß Block Null von 863 gewählt ist. Neben der Blockanzeige wird die x- und y-Position des momentan bearbeiteten Sprites gezeigt. Welches dies ist, sehen Sie über dem linken großen Sprite. In einer Reihe von »1« bis »8«, die der Sprite-Reihe in der oberen Bildschirmhälfte entspricht, ist die Nummer des aktuellen invertiert dargestellt. Daneben sehen Sie eine Auswahl von drei Farben. Eine davon ist mit einem blinkenden Rahmen versehen. Das ist die aktuelle Zeichenfarbe.

Sprite zeichnen

Mit den CRSR-Tasten bewegen Sie ein kleines weißes Quadrat, den Cursor, über das rechte Editierfeld. Wundern Sie sich nicht, wenn der Cursor auch im Feld links sichtbar wird. Zeigen beide Felder das gleiche Sprite, werden sie auch beide gleichzeitig bearbeitet. <SPACE> oder <RETURN> setzt oder löscht jeweils den Punkt unter dem Cursor (s. »DRAWMODE«).

Optionen wählen

Bewegen Sie doch einmal mit <CRSR rechts> den Cursor aus dem Editierfeld und drücken Sie <CRSR runter>. Ganz rechts in der unteren Bildschirmhälfte bewegt sich der Balken über die Befehlsauswahl. <RETURN> oder <SPACE> wählt eine von 39 Optionen, die nach Durchführung durch nochmaliges <SPACE> oder <RETURN> abgebrochen werden kann.

>1

<RETURN> oder <SPACE> blättert das gewählte Sprite (1 bis 8) um eine Blockposition weiter (0 bis 863)

>50

Hier wird um 50 Speicherpositionen aufwärts geblättert.

<1

Verringert die Speicherposition um eins.

<50

... verringert um 50.

SPRITE

<CRSR links/rechts> wählt das zu bearbeitende Sprite aus einem der acht oben dargestellten. Welche Nummer wird über dem vergrößerten Sprite links angezeigt? Vergessen Sie nicht zum Schluß mit <RETURN> oder <SPACE> zu bestätigen.

2ND SPRITE

... bestimmt den links unten vergrößert gezeigten Sprite. Auch hier wird mit den CRSR-Tasten eines der acht oben gezeigten Sprites übernommen. Das so bestimmte Sprite kann ins Editierfeld (rechtes vergrößertes Sprite) übernommen oder mit ihm logisch verknüpft werden (s. AND, OR, EOR usw.).



Der Arbeitsbildschirm zeigt acht Sprites in der oberen und die Informations- und Editierfelder in der unteren Bildschirmhälfte

ON/OFF

<RETURN> schaltet die Anzeige des aktuellen Sprites im oberen Bildschirm wechselweise ein oder aus. Das aktuelle Sprite ist immer das im Editierfeld sichtbare.

POSITION

Das aktuelle Sprite der oberen Bildschirmhälfte läßt sich per Cursor-Tasten an eine beliebige Position schieben.

EXPANSION

... vergrößert oder verkleinert das aktuelle Sprite in der oberen Bildschirmhälfte.

MULTICOLOR

Schaltet den Farbmodus für das gewählt Sprite ein bzw. aus. In diesem Modus ist ein Spritepunkt doppelt so breit wie im Einfarbmodus, kann aber vier Farben annehmen. Ist Multicolor eingeschaltet und Sie konstruieren im Editierfeld, erlaubt das Programm ein Verlassen des Feldes nach oben. Hier kann eine der drei gezeigten Farben mit <CRSR rechts/links> gewählt werden. Wenn Sie den Cursor nach unten bewegen, arbeiten Sie mit dieser Farbe weiter. Treffen Sie beim Setzen eines Punktes auf die gleiche Farbe, wird die Hintergrundfarbe verwendet. Das heißt, das Sprite ist an diesem Punkt durchsichtig (s.a. »DRAWMODE«).

SPRITECOLOR

... ändert die Farbe des aktuellen Sprites im Normalmodus. Bei Multicolor wechselt die momentan gewählte Farbe.

MULTICOLOR #1

... wechselt die Farbe Nr. 1 aller Multicolorsprites.

MULTICOLOR #2

... ändert Farbe Nr. 2 aller Multicolorsprites.

BACKGROUND #1

... wechselt die Hintergrundfarbe oben.

BACKGROUND #2

... ändert die Hintergrundfarbe unten.

FOREGROUND

... wechselt die Textfarbe.

CLR

... löscht das aktuelle Sprite. Alle Sprites verschwinden vom Bildschirm, da Sie auf denselben Speicherplatz zeigen. Fahren Sie zur Option »SPRITE«, wählen Sie »2« und ändern Sie (>1) die Speicherposition. An der Position von Sprite 2 oben erscheint ein unregelmäßiges Muster.

ALL CLR

Vorsicht vor dieser Option. Hier werden alle 864 Sprites gelöscht. Verwenden Sie diese Löschart nur nach dem ersten Programmstart, bzw. wenn Sie sichergestellt haben, daß alle Daten gesichert sind (»SAVE«).

REVERSE

... invertiert die Farben des gewählten Sprites.

ROTATE

... dreht das Sprite um 90 Grad nach links.

PUSH

Per Cursor-Tasten läßt sich der Inhalt des Sprites in die entsprechende Richtung scrollen. Vergessen Sie nicht, diese Option mit <RETURN> abzuschließen.

COPY

... erlaubt das Kopieren von Sprites. Zunächst markieren Sie das erste zu kopierende Sprite. Dazu sind die Cursor-Tasten umgeschaltet. <CRSR rechts/links> blättert im Speicher (Anzeige neben »BL:« und im Editierfeld) jeweils um ein Sprite aufwärts, bzw. abwärts. <CRSR auf/ab> blättert um 50 Sprites. Nach <RETURN> markieren Sie das letzte Sprite des Kopierbereichs und bestätigen mit <RETURN>. Danach wird noch die Wahl des Bestimmungsorts erfragt.

MIRROR

... spiegelt das Sprite in Richtung der Cursor-Tasten. Diese Option muß mit <RETURN> wieder verlassen werden.

Kurzinfo: Sprite Eddi 864

Programmart: Sprite Konstruktion und Animation
Laden: LOAD"SPRITE-EDDI 864",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 23
Programmautor: A. Stolk

Kurzinfo: Spriteripper

Programmart: holt Sprites aus Programmen
Laden: LOAD"SPR-RIPPER $1000",8,1
Starten: nach dem Laden SYS4096 eingeben
Benötigte Blocks: 5
Programmautor: A. Stolk

RETIRE

... verkleinert die Zeichnung im Editierfeld.

DIRECTORY

... bringt das Inhaltsverzeichnis der Disk auf den Screen.

DISK-AND/STAT

... erlaubt eine Abfrage des Diskettenstatus oder Übersendung eines Befehls. Nach Wahl dieser Option wird ein Eingabefeld geöffnet, sichtbar an einem < (Prompt). Wenn Sie mit <RETURN> ohne Eingabe bestätigen, liest Sprite Eddi den Diskettenstatus. Jede Texteingabe gilt als Floppybefehl.

DEVICE #8

... ändert die Gerätenummer der angesprochenen Diskettenstation. So lassen sich zwei Floppys anschließen.

LOAD

... lädt Spritefiles von Diskette. Zuerst verlangt das Programm nach der Eingabe des File-Namens. Falls Sie sich nicht sicher sind, sollten Sie vorher unter »DIRECTORY« nachsehen. Nach der Namenseingabe muß die Position, an die geladen werden soll, angegeben werden. Diese Anwahl funktioniert wie bei »COPY«.

SAVE

... speichert Sprites. Nach der Eingabe des File-Namens bestimmen Sie das erste und letzte zu speichernde Sprite.

MOVE

... startet einen Bewegungsablauf (Animation). Ein weiteres <RETURN> stoppt diesen Vorgang wieder. Zuvor sollten Sie allerdings die Parameter für »MOVE SET«, »MOVE SPEED« und Richtung eingeben.

MOVE SET

... legt fest, welche Blocks animiert werden sollen. Die Sprites einer Serie müssen lückenlos im Speicher sein.

MOVE: >

... definiert die Bewegungsrichtung. Drei Optionen gibt's:

> - Speicher zählt aufwärts. Nach Erreichen des letzten Blocks startet die Animation wieder beim ersten.

< - Speicher zählt abwärts. Nach Erreichen des ersten Blocks, startet die Animation erneut beim letzten.

<> - Speicher zählt in beide Richtungen. Zuerst wird aufwärts gezählt. Nach Erreichen des letzten Blocks wird die Zählrichtung gewechselt und abwärts bis zum ersten Block gezählt. Ab hier wieder aufwärts usw.

HARDEXPAND

... erweitert das Sprite zur Größe von vier Sprites. Die Bestimmungsposition müssen Sie angeben. Achtung: Ab dieser Position werden bei vier Sprites die Inhalte mit dem vergrößerten Sprite gefüllt. Die alten Sprites gehen verloren.

AND

... verknüpft das Editierfenster logisch UND mit einem anderen Sprite. Nach dieser Funktion sind im Editierfenster nur noch Punkte sichtbar, die in beiden Sprites übereinstimmen.

OR

Hier wird logisch ODER verknüpft, d.h. in beiden Sprites gesetzte Punkte erscheinen zusammen im Editierfenster.

EOR

Bei der Exklusiv-ODER-Verknüpfung erscheinen Punkte an gleicher Position gelöscht und leere Stellen an gleichen Positionen als Punkt.

DRAWMODE

... legt die Art des Punktezeichnens fest. Die drei Optionen:

DRAW - zeichnet den Punkt in der aktuellen Farbe.

ERASE - löscht immer (Hintergrundfarbe)

FLIP - invertiert. War vorher ein Punkt, wird gelöscht, ansonsten der Punkt gesetzt.

EXIT

... beendet das Programm nach einer Sicherheitsabfrage.

Damit Sie aber auch Sprites als Vorlagen bekommen, befindet sich der »Spriteripper« mit auf der beiliegenden Diskette. Er ermöglicht das Kopieren von Sprites aus Programmen, wenn sie sich mit RESET unterbrechen lassen und Sie einen RESET-Taster eingebaut haben.

Dazu spielen oder arbeiten Sie mit Ihrem Programm wie gewohnt, beenden aber nicht wie immer, sondern brechen mit RESET ab. Danach laden Sie:

```
LOAD"SPR-RIPPER $1000",8,1
```

und starten mit SYS4096.

In der oberen Bildschirmhälfte erscheint dann ein Spritemuster in acht Darstellungsarten. Seine Position im Speicher wird in der unteren Bildschirmhälfte (neben »Memory«) gezeigt. Mit den Funktionstasten kann im Speicher geblättert werden.

<F> markiert das erste zu speichernde Sprite, <L> das letzte. Wenn Sie danach <S> drücken, wird dieser Bereich unter einem einzugebenden Namen auf Diskette gespeichert. <X> beendet den Spriteripper. (gr)

Grafikprogrammierung

Sprites ohne Grenzen

Für Spiele sind die Sprites des C64 das Salz in der Suppe, denn diese beweglichen Minigrafiken sorgen für die Action auf dem Bildschirm, wovon die Games leben.

Der Video-Interface-Chip (VIC) verwaltet frei bewegliche Objekte, die Sprites genannt werden. Sie können unabhängig von anderen Grafiken (Charset oder Bitmap) auf dem Screen bewegt werden. Jeder Computerfreak dürfte Sprites aus den verschiedensten Spielen kennen, wo sie über den Bildschirm huschen und Spielfiguren oder Raumschiffe darstellen. Über spezielle Register (Speicherstellen) des VIC können die Sprites angesteuert werden. Die wichtigsten VIC-Register finden Sie im C64-Handbuch.

Platz für Sprites

Jedes Sprite besteht aus 63 Bytes, die in einer Matrix 3 mal 21 Byte (Breite/Höhe) angeordnet sind. Hierzu finden Sie eine Abb. in Ihrem C64-Handbuch. Sprite-Muster sind immer blockweise im Speicher abgelegt, d.h. jede 64 Byte kann eine Matrix im Speicher des C64 liegen. Das überzählige Byte (die Matrix besteht ja nur als 63 Byte) kann als Daten-Byte für die Farbe o.ä. verwendet werden. Die Form eines Sprites kann auf sehr aufwendige Art und Weise berechnet und die ermittelten Bytes in den Speicher geschrieben werden. Diese Methode wird im Handbuch des C64 beschrieben, ist aber wie gesagt sehr umständlich. Einfacher geht's mit einem Sprite-Editor. Mit diesem werden die Muster für Sprites gezeichnet und können später weiterverarbeitet werden. Einen solchen Editor finden sie in diesem Heft. Einige fertige Sprites zum Probieren enthält die beiliegende Diskette. Aber wieder zurück zu den Sprites und ihrem Platz im Speicher. Die Lage der Matrix im Speicher richtet sich nach der gewählten Grafikbank (läßt sich in CIA-Speicherstelle Hex. $dd00) umstellen). Nach dem Einschalten des C64 benutzt man automatisch Bank 1. Dabei wollen wir es auch belassen, denn für unsere Zwecke genügt die Bank 1. Als brauchbarer Speicher für unsere Sprite-Formen erweisen sich folgende Speicherbereiche:
0828- 1019/$033c-$03fb Kassettenpuffer
2048-16320/$0800-$3fc0 Basic-Speicher.

```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
POKE53248,150
READY.
POKE53249,100
READY.
POKE53269,1
READY.
POKE2040,255
READY.
LOAD"DEMO-SPRITES",8,1
SEARCHING FOR DEMO-SPRITES
LOADING
READY.
```

[1] Mit vier POKEs ein Sprite auf den Bildschirm

Beim zweiten Bereich gibt es aber eine Einschränkung! Der Bereich von 4096-8192 (hex.$1000-$2000) ist für Sprites tabu, da dort der Zeichensatz des C64 gespiegelt wird. Würden dort Sprite-Daten liegen und man teilt dem C64 mit, daß ein Sprite die Form aus diesem Speicherbereich annehmen soll, würde nur Grafikmüll auf dem Bildschirm erscheinen.Da bei einem Reset der Kassettenpuffer gelöscht wird und die Grafikdaten futsch sind, scheidet dieser Bereich für unsere Experimente aus. Natürlich kann der Bereich benutzt werden, aber man riskiert bei einem Absturz des Computers den Verlust der Daten. Der Basic-Start dürfte für alle Programmierer, die die eingebaute Hochsprache benutzen, zum Problem werden. Abhilfe schafft da das Hochsetzen des Basic-Starts (s. 64'er 12/92 Basic-Corner). Wir benutzen für unsere Demos den oberen Bereich und dort nur 254 Byte für vier Sprites. Im Speziellen ist das der Bereich von 16128 bis 16320 (hex.$ 3f00-$3fc0). Kürzere Basic-Programme überschreiben diesen nicht und sind somit nicht gefährlich. Assembler-Programmierer, die diesen Exkurs in die Welt der Sprites verfolgen, können einen beliebigen geeigneten Speicherbereich benutzen. Aber immer daran denken, daß sich Sprite-Daten und Programm nicht überschneiden dürfen. Die Grundlagen der Sprite-Programmierung werden mit Hilfe von Basic-Befehlen vermittelt. Für Maschinensprache-Freaks dürfte es kein Problem sein, diese Befehle in Assembler zu übertragen. Unsere Daten für die Sprite-Formen liegen also in den Blöcken 252 bis 255 der ersten Grafikbank des C64. Jetzt die Frage: Wieso 252 bis 255? Auf die Banknummer kommt man nach folgendem Muster: **Speicherbereich/64=Blockzahl - 16192/64=253**. Die Blocknummer ist eine wichtige Konstante, denn jedes Sprite hat ein Register, wo die Nummer eingetragen und so die Form des Sprites zugewiesen werden kann. In Tabelle 1 finden Sie die acht Adressen. Will man beispielsweise Block 252 dem Sprite zuweisen, genügt folgender Befehl:

```
POKE 2040,252: REM SPRITEFORM
```

Schwups und schon hat das Sprite die angegebene Form. Nachdem Sie den Befehl im Direktmodus eingegeben haben, werden Sie sich sicher wundern, daß nichts passiert ist. Das hat einen einfachen Grund – Das Sprite 1 ist noch gar nicht auf dem Bildschirm. Wie wir das bewerkstelligen, zeigt das nächste Kapitel.

Sprites zeigen

Um ein Sprite auf dem Bildschirm zu zeigen, muß der Programmierer drei wesentliche Schritte ausführen:
1. Sprite positionieren
2. Farbe zuweisen
3. Sprite einschalten.

Also beginnen wir erst einmal beim ersten Punkt. Jedem der acht Sprites kann eine Position auf dem Bildschirm zugewiesen werden. Der Platz auf dem Bildschirm wird über eine X- und Y-Koordinate bestimmt. Von 53248 bis 53263 befinden sich die Koordinaten aller acht Sprites. Wir wollen uns weiter mit dem ersten Sprite beschäftigen. Die Koordinaten werden in die Register 1 und 2 des VIC (Startadresse 53248/$d000) eingetragen. Dies erfolgt mit den POKES:

```
POKE 53248,100:REM SPRITE 1 X-POS.
POKE 53249,100: REM SPRITE 1 Y-POSITION
```

Der Wertebereich der Koordinaten beträgt 0 bis 255. Wenn wir das Sprite vollständig auf dem Bildschirm haben, können Sie mit anderen Werten ein wenig experimentieren. Der Nullpunkt des Koordinatensystems für die Sprites liegt im linken oberen Eck des Bildschirms unter dem Rahmen. Auf diese Weise können Sprites unter dem Bildschirmrahmen ins Bild hineinbewegt werden. Nun aber zu Punkt zwei, der Farbe der Sprites. Die Farben für die acht Sprites werden in die Adressen 53287 bis 53294 (hex. $d027-$d02e) geschrieben, ähnlich wie man es beim Einfärben des Bildschirms und des Rahmens handhabt. Die Farbcodes sind dabei analog. Wir wollen unser Sprite weiß färben:

```
POKE 53287,1: REM SPRITEFARBE (WEISS)
```

Sprite-Blockzeiger (Tabelle 1)

	1	2	3	4	5	6	7	8
Speicherstelle (dez.)	2040	2041	2042	2043	2044	2045	2046	2047
(hex.)	$07f8	$07f9	$07fa	$07fb	$07fc	$07fd	$07fe	$07ff

Weitere Sprite-Artikel

Trickfilm wie im Kino	7/91 Seite 12
Sprite-Multiplexen	7/91 Seite 72
Sprites im Sideborder	4/92 Seite 58
Sprites in Assembler	7/92 Seite 52
Sprite-Move	2/92 Seite 63

Als dritten und letzten Punkt schalten wir das Sprite ein. Für diese Aktion wird das Register 53249 (hex. $d015) benutzt. Die Bits in dieser Speicherstelle symbolisieren je ein Sprite. Wird dieses Bit gesetzt, bedeutet es, daß dieses Sprite eingeschaltet ist. Wir schalten also das Sprite 1 ein und POKEn deshalb:

```
POKE 53249,1: REM SPRITE 1 EIN
```

Nun müßte ein Sprite auf dem Bildschirm erscheinen, das aber noch aus Grafikmüll besteht. Um dem Ganzen eine gute Form zu geben, muß die entsprechende Grafik-Speicher stehen. Zeichnen Sie sich mit dem Sprite-Editor in diesem Heft eine Form Ihrer Wahl und sicheren Sie das Ganze auf Diskette. Die Daten müssen dann direkt an Speicherstelle 16192 (hex.$ 3f40) geladen werden. Dazu benutzen Sie am besten einen Maschinensprache-Monitor. Wenn Sie nicht über entsprechende Kenntnisse verfügen, können Sie auch die Demo-Sprites von Diskette nutzen. Sie laden Sie mit:

```
LOAD"DEMO-SPRITE",8,1
```

Danach müßte das Sprite die Form wie in Abb. 1 haben. Nun können Sie ein wenig mit den bekannten Registern experimentieren. Verändern Sie doch einfach die Farbe des Sprites oder die Koordinaten.

Sprite-Bewegung

Um die Sprites auf dem Bildschirm zu bewegen, müssen nur die Koordinaten zyklisch verändert werden. Dazu gibt es mehrere Möglichkeiten. Eine FOR-NEXT-Schleife ist zur Demonstration am besten geeignet.

```
10 FOR X=0 TO 255: POKE 53248,X: NEXT
```

Diese Schleife bewegt das Sprite 1 in der Horizontalen von der Position 0 bis 255. Das funktioniert natürlich auch vertikal. Startet man in einem Basic-Programm mit einem GOTO-Sprung immer wieder, läuft das Sprite ständig von links nach rechts. Nun wird sich der aufmerksame Leser wundern, daß das Sprite mitten im Bild verschwindet, als wäre da der Bildschirmrahmen. Dieses Phänomen ist leicht zu erklären. Die vertikalen Koordinaten für die Sprites reichen von 0 bis 255, die horizontalen jedoch gehen darüber hinaus. Da eine 8-Bit-Zahl bekanntlich als größten Wert nur 255 annehmen kann, gibt es für jedes Sprite ein neuntes Bit, welches sich im Register 53264 (hex.$d010) befindet. Die Belegung ist analog dem Register für das Anschalten des Sprites (s.o.). Als Faustregel gilt: Soll das Sprite auf dem rechten Bildschirm weiterlaufen, muß das Bit im Register 53264 für das Sprite gesetzt sein. Wollen wir also Sprite 1 über den ganzen Bildschirm gleiten lassen, muß bei Position 255 das Bit im Register 53264 gesetzt werden und die Koordinate wieder 0 betragen. Das Sprite ist jetzt in der echten Seite (gesetztes Bit) auf Position 0. Um unsere obere Schleife perfekt zu gestalten, fügen wir eine Zeile ein:

```
10 FOR X= 0 TO 255: POKE5248,X: NEXT: REM SPRITE
BEWEGEN
20 RL=1-RL: POKE 53264:RL: BIT SETZEN BZW. LOESCHEN
30 GOTO 10: REM WIEDER NACH OBEN
```

Will man mehrere Sprites zugleich benutzen, wird die Angelegenheit etwas komplizierter. Unter Umständen ist dann Basic zu langsam und man muß die Sprite-Programmierung in Assembler fortsetzen. Es gibt aber genügend Programmiertricks in Basic, um zufriedenstellende Ergebnisse zu erzielen. Grundkenntnisse im Handling von Bits und die Benutzung der Verknüpfungsbefehle (AND und OR) sind dabei Voraussetzung. Maschinensprache-Programmierer, mit einem Assembler oder Monitor bewaffnet, die diesen Artikel verfolgen, müssen bei den vorgestellten Beispielen zur Bewegung von Sprites natürlich Pausen einbauen, da in Assembler alles rasend schnell geht.

Farbe, Form und Priorität

Leser, die ein wenig mit den Pointern für die Sprite-Form experimentiert haben und wissen wollten, was sich in den Blöcken 254 und 255 befindet, werden beim Umschalten auf diese Speicherbereiche Grafikmüll auf dem Bildschirm entdeckt haben. Hier haben wir aber keinen Fehler beim Zeichen des Sprites gemacht, sondern sogenannte Multicolor-Sprites auf den Bildschirm zeigen wollen. Multicolor heißt, daß neben der Sprite-Farbe und dem Hintergrund (der durchscheint, wenn kein Punkt gesetzt ist) noch zwei weitere Farben für die Sprites verwendet werden können. Kleiner Nachteil: die Auflösung in der Horizontalen wird halbiert und die Pixel werden immer paarweise dargestellt. Um diesen Modus für ein Sprite einzustellen, muß in Register 53276 (hex.$d01c) das entsprechende Bit gesetzt werden. Wenn Sie nun Sprite 1 den Block 254 zuweisen und dann mit:

```
POKE53276,1
```

den Multicolor-Modus einschalten, sieht das Bild viel freundlicher aus und es kommt Farbe ins Spiel. Die beiden zusätzlichen Farben gelten für alle Sprites und werden in den Registern 53285 (hex. $d025) und 53286 (hex. $d026) eingetragen. Die Farbcodes entsprechen den bekannten Codes für den C 64 (z.B. Bildschirmfarbe). Eine weitere Eigenschaft der Sprites besteht darin, daß man sie in X- und Y-Richtung vergrößern kann. Beschreibt man dazu das entsprechende Bit in den Registern, wird das Sprite in X- oder Y-Richtung doppelt so groß. Die Auflösung nimmt ebenfalls zu. Mit dem Multicolor- und den Vergrößerungsregistern kann nun ein wenig herumgedoktert werden. Nun zur Priorität der Sprites. Wenn man nämlich Sprites übereinanderlegt, verdeckt ein Sprite das andere. Hier gilt, daß ein Sprite mit einer niederen Nummer immer vor einem Sprite mit höherer Nummer liegt, wenn sie dieselben Positionen haben. Dieser Fakt ermöglicht sogenannte Overlay-Sprites, d.h. man kann eine Figur auch mit zwei Sprites gestalten (z.B. ein Multicolor-Sprite und ein Sprite darübergelegt in Hires). Hier ergeben sich zahlreiche Kombinationsmöglichkeiten, die optisch sehr anspruchsvolle Grafiken ermöglichen (s. Listing auf Disk). Außerdem kann in Register 53275 festgelegt werden, ob das Sprite vor einer Grafik (Zeichensatz oder Bitmap) erscheinen soll. Ist das entsprechende Bit für das Sprite in dieser Speicherstelle gesetzt, wird es hinter die Grafik gesetzt.

Kollision

In Spielen ist es sehr wichtig zu wissen ob das Heldensprite mit einem Gegner zusammengestoßen ist und dem Spieler ein Leben abgezogen wird. Eine Möglichkeit ist die Überprüfung der Koordinaten der Sprites. Diese Methode erfordert aber ein gewisses Maß an Rechenzeit, die man vielleicht für andere Programmteile benötigt (ich denke da vor allem an das langsame Basic). Es gibt eine viel elegantere Abwicklung dieses Punkts. Dazu hat der VIC auch ein Spezialregister, das dem Programmierer die Arbeit abnimmt. Das Register 53278 (hex. $d01e) ist für diese Überprüfung verantwortlich. Stoßen zwei Sprites zusammen, wird im Register 53278 das entsprechende Bit gesetzt. Kollidieren beispielsweise Sprite 2 und 6, werden die entsprechenden Bits gesetzt und im Register 53278 steht 34 (bin. %00100010). Mit folgender Programmzeile kann man einfach überprüfen ob es eine Kollision gab:

```
10 IF PEEK(53278)<> 0 THEN PRINT "KOLLISION"
```

Um herauszufinden welche Sprites kollidieren muß das Ergebnis ausmaskiert werden. Das Register 53279 (hex. $d01f) ist für den Zusammenstoß zwischen Hintergrundgrafik und Sprites verantwortlich. Es arbeitet wie das Register für die Sprite-Sprite-Kollision, kann aber für die Arbeit vernachlässigt werden, da es zu ungenau ist. In Maschinensprache lassen sich nur befriedigende Ergebnisse erzielen und in Basic versagt diese Methode. Um eine Kollision zu prüfen, muß man sich eine eigene Routine programmieren, die die Position des Sprites ermittelt und überprüft, ob an dieser Stelle Hintergrund auf dem Bildschirm ist. Das geht selbstverständlich nur in Maschinensprache.

Sprites und Assembler

Wie schon in den vorangegangenen Kapiteln erwähnt, kann man nur in Maschinensprache schnelle Sprite-Routinen programmieren, ohne daß die Bytes flimmern und deren Bewegung stottert. Eine Beispielroutine dafür finden Sie in Listing. Dort wird Sprite 1 auf den Bildschirm gebracht (X- und Y-Koordinate=100) und eine Interruptroutine initialisiert. Danach setzt man das Sprite 2 auf den Bildschirm, welches nach einem Tastendruck auf dem Bildschirm erscheint. Das Sprite 1 bewegt sich von rechts nach links. Nachdem das Sprite 2 auf dem Bildschirm erscheint und mit Sprite 1 zusammenstößt, wechselt das erste Sprite die Richtung. Die Interruptroutine synchronisiert die Bewegung des Sprites 1 mit dem Rasterstrahl und schließt das Flimmern des Sprites während der Bewegung aus. Das Demo auf Diskette wird mit SYS 4096 gestartet. Zur Vertiefung dieses Themas finden Sie einige weitere aufschlußreiche Artikel in unseren 64'er-Stammheften (s. Textkasten). Das gleiche gilt für das Multiplexen von Sprites (Darstellung von mehr als acht Sprites auf dem Bildschirm) und Manipulieren von Sprite-Formen. Für solche Themen ist unsere Proficorner im Stammheft zu empfehlen. (lb)

```
start     lda #00         ;rahmen und
          sta $d020       ;bildschirm
          sta $d021       ;schwarz
          lda #$01        ;sprite 1
          sta $d027       ;weiss faerben
          lda #$64        ;sprite 1
          sta $d000       ;x- und y-koor.
          sta $d001       ;auf 100
          lda #$fd        ;sprite 2 form
          sta $07f8       ;aus block 253
          lda #$1         ;sprite 1
          sta $d015       ;einschalten
          jsr form
irqinit   sei             ;interrupt sperren
          lda #<ineu      ;neue irq-routine
          ldx #>ineu      ;initialisieren
          sta $314
          stx $315
          lda #%10000001
          sta $d01a
          lda #$20
          sta $d012
          cli             ;interrupt freigeben
          lda #$64        ;sprite 2
          sta $d002       ;x- und y-
          sta $d003       ;koor. auf 100
          lda #$02        ;sprite 2
          sta $d028       ;rot faerben
          lda #$fd        ;sprite 2 form
          sta $07f9       ;aus block 253
key       jsr $ffe4       ;auf taste
          beq key         ;warten
          lda #$03        ;sprite 2 zu
          sta $d015       ;sprite 1 zuschalten
          rts             ;zurueck zu basic
ineu      lda $d019
          sta $d019       ;register loeschen
          bmi raster      ;raster-irq?
          lda $dc0d       ;nein=>irq-reg.
                          ;loeschen
          cli             ;irq frei
          jmp $ea31       ;zu timer-irq
raster    lda $d01e       ;kollision pruefen
          beq loop        ;nein zu loop
          inc richtung    ;ja richtung aendern
loop      lda richtung    ;richtung
          and #1          ;pruefen
          bne left
right     jsr pruef1
          inc $d000       ;sprite-koor.
                          ;erhoehen
          jmp $ea7e       ;zurueck zu
                          ;system
left      jsr pruef2      dec $d000
                          ;sprite-koor.
                          ;erhoehen
          jmp $ea7e       ;zurueck zu
                          ;system
pruef1    lda $d000       ;sub-routine fuer
          cmp #$ff        ;das setzen/loeschen
          bne back1       ;des 9 bit der
          inc va          ;x-koordinate
          lda va          ;des sprites1
          and #$01
          sta $d010
          lda #$00
back1     rts
pruef2    lda $d000       ;sub-routine fuer
          bne back2       ;das setzen/loeschen
          inc va          ;des 9 bit der
          lda va          ;x-koordinate
          and #$01        ;des sprites1
          sta $d010
back2     rts
richtung  .byte 0va       .byte 0
form      ldy #$40        ;spriteform
          lda #$ff        ;nach block 253
looping   sta $3f40,y     ;schreiben (kasten)
          dey
          bpl looping
          rts
```

Listing

SONDERHEFT

VORSCHAU



Nr. 88 gibt's ab 19.03.93 bei Ihrem Zeitschriftenhändler

Ein Computersystem ohne Drucker ist wie eine Flasche Spätlese, zu der man den Korkenzieher verloren hat. Textverarbeitungs-, Datenverwaltungs- und Grafikprogramme z.B. sind ohne Druckausgabe nur die Hälfte wert. Unser nächstes Sonderheft ist allen C-64-Anwendern gewidmet, denen problemlose Zusammenarbeit von Druckern mit den verschiedensten Softwareprodukten am Herzen liegt.

Unsere Highlights:

■ Der C64 als Zeitungsmacher: »Giga-Publish«, das komplette DTP-Paket in neuer, überarbeiteter Auflage. Unser Workshop führt Sie Schritt für Schritt in die Welt des Desktop-Publishing ein: Seiten für Schüler-, Vereins- oder Clubzeitungen lassen sich künftig im Handumdrehen entwerfen und als fertiges Dokument zum Grafikdrucker schicken.

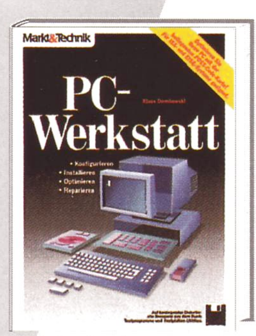
■ Tips & Tools zur Druckeranpassung – damit läßt sich fast jedes Gerät überreden, Daten korrekt auszugeben.

Aus aktuellen oder technischen Gründen können Themen ausgetauscht werden. Wir bitten um Ihr Verständnis.

Sichern Sie sich professionelles HARDWARE-WISSEN!

An alle Hardware-Einsteiger und Hardware-Profis: Die Experten des Markt&Technik-Verlages bieten Ihnen ihr Wissen an. Grundlegend und umfassend - professionell und verständlich. Vom Standard-Werk der Hardware über ein aufregendes Multimedia-Buch bis hin zur detaillierten Anleitung zum Aufbau eines Mini-Rechners. Sichern Sie sich genau das Hardware-Wissen, das Sie brauchen. Denn nur wer die Hardware seines PCs kennt, kann wirklich effizient arbeiten!

Holen Sie sich wertvolles Hardware-Wissen: Buch + Diskette + Platine!



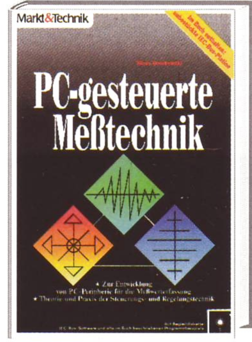
Klaus Dembowski
PC-WERKSTATT
Der Leser erfährt hier alles über die Systemkomponenten des PC, so daß jede Scheu vor dem Kontakt mit der Hardware entfällt. Speicheraufrüstung oder ein zusätzliches Laufwerk sind damit kein Problem mehr. Und wenn der Rechner überhaupt nicht mehr funktioniert, kann man sich mit Hilfe der beiliegenden Platine eine Post-Code-Karte zur Fehlerdiagnose selber bauen.
1992, ca. 300 Seiten, inkl. Diskette und Platine
ISBN 3-87791-344-X ca. DM 98,-



P. Wratil/D. Schwampe
MULTIMEDIA FÜR VIDEO UND PC
Einladung zur Entdeckungsreise ins Multimedia-Land. Wie Sie die aufregenden Möglichkeiten der Technik selbst erleben können, beschreiben die Autoren in diesem Praxisbuch, das eine unbestückte Platine enthält. Mit etwas Lötzinn und Erfahrung läßt sich daraus eine Genlock- und Multimedia-Karte basteln. Ein Multimedia-Software-Paket und Diagnose-Software werden auf Diskette mitgeliefert.
1992, 372 Seiten, inkl. Diskette und Platine
ISBN 3-87791-194-3 DM 98,-



Herbert Bernstein
PC-SPEICHERMEDIEN
Alles über Halbleiterspeicher, Floppy- und Festplatten-Laufwerke, Backup-Systeme, optische Speicher und deren Schnittstellen. Ein technischer Wegweiser und Einkaufsführer für fortgeschrittene PC-Anwender, die ihren Computer eigenhändig erweitern wollen. Neben ganz praktischen Kauf- und Einbau-Tips erfährt der Leser eine Menge über die Funktion der Bauteile.
1992, ca. 800 Seiten, inkl. Diskette
ISBN 3-87791-162-5 ca. DM 89,-



Klaus Dembowski
PC-GESTEUERTE MESSTECHNIK
Die praxisgerechte Anleitung zur Entwicklung von Meßsystemen mit Hilfe von Einsteckkarten, der RS232- und der IEC-Schnittstelle. Einsteigern und Profis liefert dieses „Kochbuch" neben erprobten Konzepten und Hintergrundwissen als Besonderheit eine IEC-Bus-Platine, die voll kompatibel zum Industriestandard ist. Die Software für die Platine und alle im Buch beschriebenen Meßprogramme werden auf einer Diskette mitgeliefert.
1991, 471 Seiten, inkl. Diskette und Platine
ISBN 3-89090-958-2 DM 119,-



Hans-Joachim Blank
LOGIKBAUSTEINE - GRUNDLAGEN, PROGRAMMIERUNG UND ANWENDUNG
Dieses vielseitige Praxisbuch beschreibt vor allem die Programmierung der modernen Logikbausteine. Und es stellt viele interessante Anwendungsbeispiele vor, z.B. Paßwortdecoder, Aufzugssteuerung oder Multibus-Schnittstelle. Auf zwei HD-Disketten (5,25") werden als Starter-Kit mitgeliefert: Logik-Compiler, Simulator für die Bausteinfunktionen, GAL-Programmer-Software und Designbeispiele!
1992, 421 Seiten, inkl. 2 Disketten
ISBN 3-87791-072-6 DM 79,-

Außerdem lieferbar:

Herbert Bernstein • **HARDWARE-HANDBUCH FÜR PC/XT/AT UND KOMPATIBLE** • 1990, 431 Seiten • ISBN 3-89090-913-2 DM 79,-

J. Koch/M. Schusser • **PC/XT/AT-KOMPENDIUM** • 1989, 504 Seiten, inkl. Diskette • ISBN 3-89090-778-4 DM 69,-

Kai Hamann • **PC-BASTELBUCH** • 1990, 309 Seiten, inkl. Diskette und Platine ISBN 3-89090-331-2 DM 98,-

Herbert Bernstein • **PC-TUNING** • 1991, 552 Seiten, inkl. Diskette ISBN 3-89090-950-7 DM 69,-

Uwe Gerlach • **DAS TRANSPUTERBUCH** • 1991, 464 Seiten, inkl. Diskette und Platine • ISBN 3-87791-019-X DM 119,-

P. Wratil/R. Schmidt • **DER PC ALS INTELLIGENTE SCHALTZENTRALE** 1990, 506 Seiten, inkl. Diskette und Platine • ISBN 3-89090-651-6 DM 119,-

H.-J. Blank/H. Bernstein • **PC-SCHALTUNGSTECHNIK IN DER PRAXIS** 1990, 506 Seiten, inkl. Diskette und Platine • ISBN 3-89090-914-0 DM 119,-

P. Wratil/R. Schmidt • **PC/XT/AT - MESSEN, STEUERN, REGELN** 1987, 255 Seiten, inkl. Platine • ISBN 3-89090-477-7 DM 99,-



DAS ERFOLGS-PROGRAMM FÜR IHR PROGRAMM!

Dies ist nur ein kleiner Ausschnitt aus dem neuen Gesamtprogramm des Markt&Technik-Verlages: Mehr als 500 Problemlösungen zu Hard- und Software warten auf Sie - jetzt bei Ihrem Buchhändler, im PC-Fachhandel und in den Computer-Abteilungen der Warenhäuser!

6221-1

